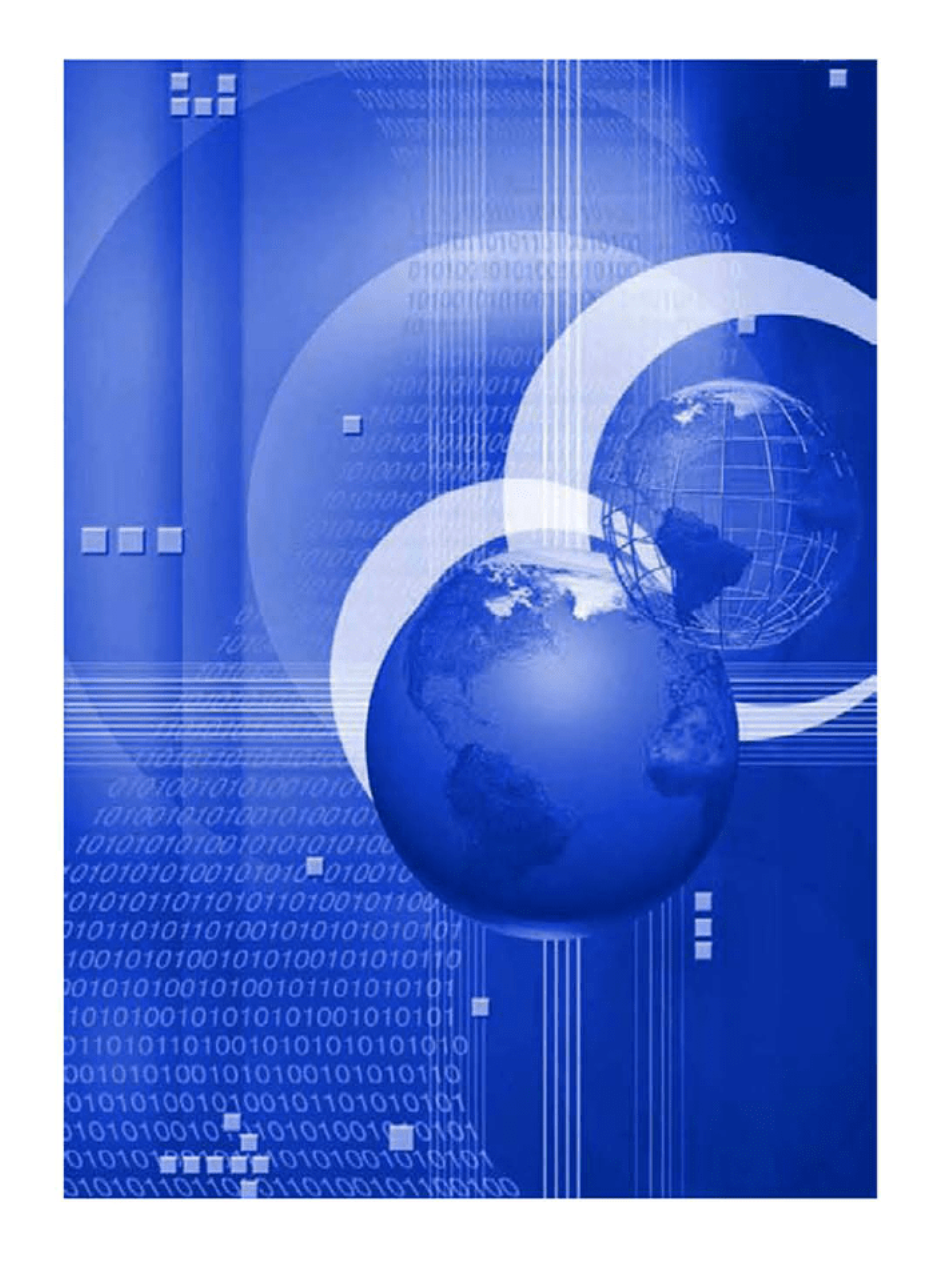
高等学校计算机应用规划教材

ASP.NET 3.5 网站开发实例教程

- ASP.NET技术
- VS 2008与VWD 2008
- XHTML标记语言
- ASP.NET服务器控件
- 页面布局
- ADO.NET数据访问
- 数据绑定
- ASP.NET AJAX
- Web服务
- XML技术



陈伟 卫琳 主编
唐国良 周飞菲 副主编

清华大学出版社

高等学校计算机应用规划教材

ASP.NET 3.5 网站开发实例教程

陈伟　卫琳　主编

唐国良　周飞菲　副主编

清华大学出版社

北　京

内 容 简 介

本书全面翔实地介绍了 ASP.NET 3.5 网站开发方法和 Web 应用程序的技术。全书共分 12 章，主要内容包括：ASP.NET 与 VWD 2008 开发平台、XHTML 基础知识、ASP.NET 基础知识、ASP.NET 服务器控件、页面切换与导航、页面布局、ADO.NET 数据访问、ASP.NET 中的数据绑定、ASP.NET AJAX、Web 服务、在 ASP.NET 中使用 XML 以及综合开发实例。此外，每章均配有实验，有助于读者对所介绍知识的理解与掌握。

本书结构清晰，实例丰富，图文对照，浅显易懂，既可作为高等院校 ASP.NET 开发课程的教材，还可作为有一定的面向对象编程基础和数据库基础，想利用 VS 2008 开发 Web 应用程序的开发人员的参考资料。

本书的电子教案、实例源代码和实验答案可以到 http://www.tupwk.com.cn/downpage/index.asp 网站下载。

图书在版编目(CIP)数据

编著. —北京：清华大学出版社，2007.7

(高等学校计算机应用规划教材)

ISBN 978-7-302-

Ⅰ. 中…　Ⅱ. ①…②…　Ⅲ.

Ⅳ. TP391.

中国版本图书馆 CIP 数据核字(2007)第 0　　　号

责任编辑： 胡辰浩(huchenhao@263.net)　袁建华

装帧设计： 孔祥丰

责任校对： 成凤进

责任印制：

出版发行： 清华大学出版社　　**地　址：** 北京清华大学学研大厦

http://www.tup.com.cn　　**邮　编：** 100084

社 总 机： 010-62770175　　**邮　购：** 010-62786544

投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn

质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：

装 订 者：

经　销： 全国新华书店

开　本： 185×260　**印　张：** 20.5　**字　数：** 512 千字

版　次： 2009 年 月第 1 版　**印　次：** 2009 年 月第 1 次印刷

印　数： 1~5000

定　价： 30.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：

前　　言

ASP.NET 技术是目前开发 Web 应用程序最流行和最前沿的技术，ASP.NET 3.5 是目前 ASP.NET 的最新版本。早在 ASP.NET 1.0 版本于 2000 年发布时，许多人就认为它在 Web 应用程序开发方面迈出了具有革命性的一步。而后来的 ASP.NET 2.0 更激动人心、更富有革命性，ASP.NET 3.5 则继续朝着这个方向前进，为在 Web 上建立应用程序提供了目前最佳的框架。

本书以易学易用为重点，充分考虑实际开发需要，用大量的实例，引导读者掌握 ASP.NET 网站开发的方法与技巧，使读者学习本书后可以立即设计出实用的 Web 应用程序。本书配有电子教案、所有实例文件和实验的源代码，即便于教师教学，也便于学生学习，具有很强的实用性。

实例和实验服务器端代码选用 C#语言编写，数据库以 SQL Server 2005 为平台。所有实例和实验的源码均在 VS 2008 环境中运行通过。

本书共分 12 章，第 1 章介绍 ASP.NET 与 VWD 2008 开发平台；第 2 章介绍 XHTML 基础知识；第 3 章介绍 ASP.NET 基础知识；第 4 章介绍 ASP.NET 服务器控件的使用方法；第 5 章介绍页面切换与导航；第 6 章介绍页面布局；第 7 章介绍 ADO.NET 数据访问；第 8 章介绍 ASP.NET 中的数据绑定方法；第 9 章介绍 ASP.NET AJAX 技术；第 10 章介绍 Web 服务；第 11 章介绍在 ASP.NET 中使用 XML 方法；第 12 章介绍基于 SQL Server 2005 数据库的综合开发实例。

本书第 1、2 章由周飞菲编写；第 3 章由李鑫编写；第 4、10 章由唐国良编写；第 5 章由申杰编写；第 6 章由李文泽编写；第 7、12 章由陈伟编写；第 8 章由赵欣编写；第 9、11 章由卫琳编写。本书由陈伟、卫琳主编，唐国良、周飞菲副主编，石磊主审。王文凡、赵卓、赵少林、程杰、张西芝、乔木等参与了部分资料的收集整理工作，在此向他们表示感谢。

本书可以作为高等院校 ASP.NET 开发课程的教材，也可以作为有一定的面向对象编程基础和数据库基础，想利用 VS 2008 开发 Web 应用程序的开发人员的参考资料。

在编写本书的过程中参考了相关文献，在此向这些文献的作者深表感谢。由于时间较紧，书中难免有错误与不足之处，恳请专家和广大读者批评指正。我们的信箱：huchenhao@263.net，电话是：010-62796045。

作　者

2009 年 1 月

目　　录

第1章　ASP.NET与VWD 2008 开发平台

本章介绍了网站建设的基本原理、流程和创建网站的工具——ASP.NET 技术的基本概况。作为一种新型的 Web 开发技术，ASP.NET 基于 Microsoft 公司的.NET 框架，支持 C# 和 VB.NET 语言，是现在主流的网站开发平台。通过本章的学习，读者能够了解如何安装、使用 ASP.NET 的集成开发环境——Visual Web Developer 2008，并能够建立简单的动态网站和页面。

本章的学习目标

- 掌握静态网站与动态网站的概念、特点
- 了解 ASP.NET 的发展历史、特点及开发环境
- 掌握安装 ASP.NET 的开发环境 VWD 2008 的步骤
- 了解开发动态网站的一般流程
- 能够创建简单的动态网站

1.1　网站建设基础知识

互联网的快速发展给人们的工作、学习和生活带来了重大变化，人们可以利用网络处理数据、获取信息，极大地提高了工作效率。在互联网开发所涉及的众多技术中，最为关键的技术之一就是网站建设技术。通过本节的学习，读者将了解在网站制作的过程中常常会见到的静态网站、动态网站等一些基本概念。

1.1.1　静态网站

传统的网站一般是采用静态网页技术制作的静态网站。在静态网站中所有的内容以 HTML 语言编写，存储在静态网页文件中，文件扩展名为：.htm、.html、.shtml、.xml 等。网页上可以出现 GIF 动画、FLASH 动画、滚动字幕等动态视觉效果，但这些动态效果只是视觉上的。这里所讨论的静态网站中的“静”是指网页内容在用户发出请求之前就已经生成了(这就是用户每次总能看到相同页面的原因)，Web 服务器只负责保存和传递 HTML 文件，

而不进行额外处理，用户只能阅读网站所提供的信息，如图 1-1 所示。



图 1-1　静态网站模型

静态网站中网页的内容相对稳定，不需要通过数据库工作，对于 Web 服务器来说，处理负担不大，因此静态网站具有容易被搜索引擎检索、访问速度比较快的优点。

静态网站的致命弱点是不易维护，为了不断更新网页内容，网站管理者必须不断地重复制作 HTML 文档，随着网站内容和信息量的日益增长，维护工作将变得十分艰巨。因此，静态网站往往适用于数据不多，网页比较固定，更新不频繁的情况。更新较少的展示型网站一般采用静态网站技术搭建。

1.1.2　动态网站

在静态网站中用户可以阅读、发布大量信息，但如果用户想拥有自己的聊天室——享受与朋友在一起自由讨论的乐趣，建立网上商店——实现自己的创业梦想……就必须采用动态网站技术进行交互式 Web 体验。

什么是动态网站呢？所谓“动”，并不是指网页上的 GIF 等动画图片，而是指用户与网站的互动性。

动态网站的概念现在还没有统一标准，但一般应满足以下几点特征。

1. 交互性

动态网站中的网页会根据用户的要求和选择而改变和响应。网站管理员只需要掌握计算机基本操作方法，就可以方便、及时地更新网站内容；浏览网站的用户可以在网站中进行查询、留言等操作。可见，动态网站技术大大增加了管理员与网站、客户与网站的互动性。

2. 通过数据库进行架构

在动态网站中，网络管理员除了要设计网页视觉效果，还要设计数据库和程序代码来使网站具有更多自动的和高级的功能。例如，购物网站中含有大量的商品种类和数量信息，为了方便查找，就应搭建数据库平台在网页上实现自动搜索。现在广泛使用的网上交易系统、在线采购系统、商务交流系统等都是由数据库提供技术支持的。

3. 在服务器端运行，方便更新

在服务器端运行的程序、网页、组件，会随不同客户、不同要求返回不同的页面，网站管理员无须手动更新网页文档，可以大大节省网站管理的工作量，如图 1-2 所示。

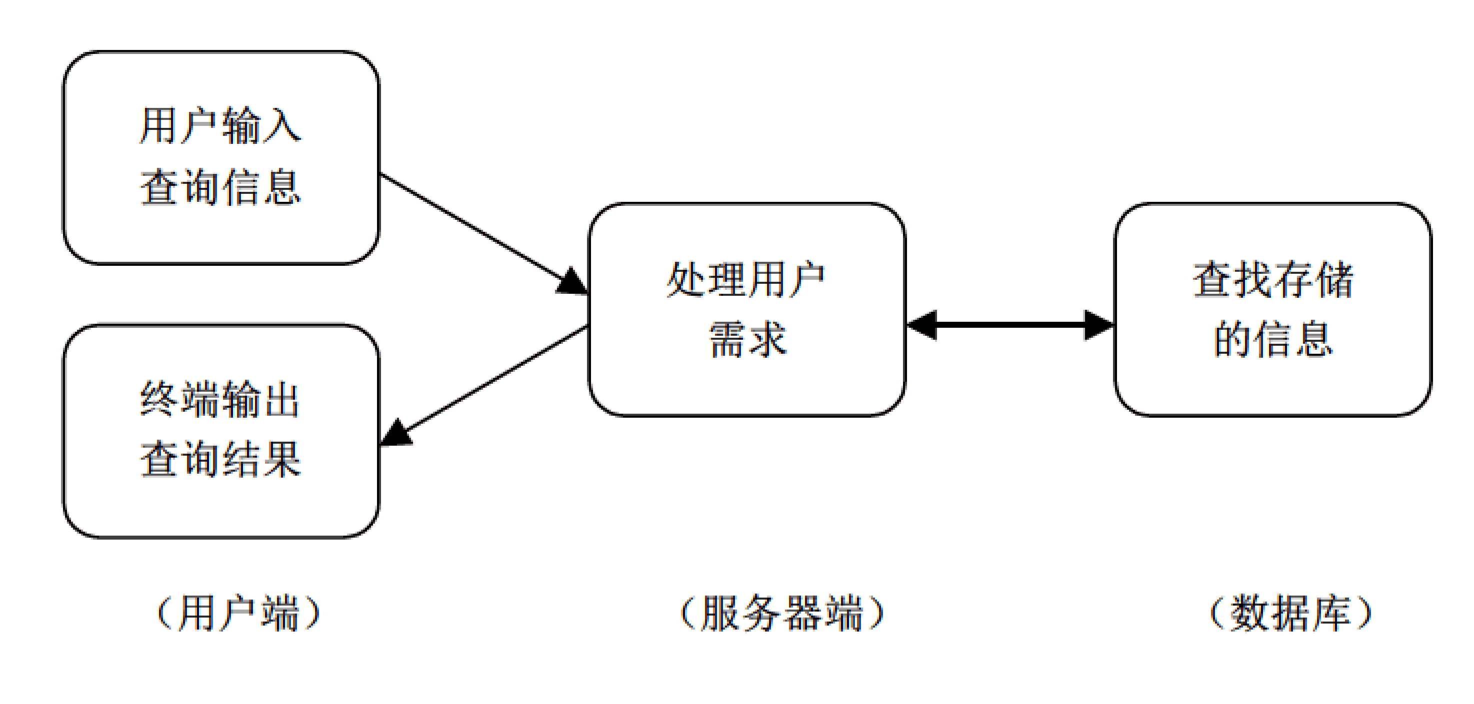


图 1-2　动态网站模型

由上述特征可以看出，静态和动态内容的主要区别在于：静态网站内容是在用户发出请求之前预先生成的，而动态网站内容则是在用户发出请求之后产生的。

动态网站在发出请求之后生成内容有两个明显的优点。

首先，服务器端可以根据用户提交的请求、用户提供的请求的内容值设置它所生成的页面内容。如：在一个电子商务网站提交用户名和密码，那么用户将看到的下一个页面就是动态生成的页面，它包含了用户的私有账户的信息。

其次，服务器端可以根据最新的可用信息设置它所生成的页面内容。如很多网站都有正在使用这个网站的用户数量的显示。用户数量的值是实时信息，是在 Web 服务器接收用户请求时获取的。

可以说，动态网站的 Web 服务器不仅负责传递网站文件，它更是一个完成信息处理的执行平台。例如一个购物网站，由两部分组成，一是用户部分(也叫前台客户端)，一是管理员部分(也叫后台服务器端)。前台客户端提供的功能有：新用户注册，已注册用户的登录，用户对商品的查询、浏览，用户对商品的在线购买等等。当一个新用户在前台注册成功，后台服务器就必须在数据库的用户表中添加相应的用户信息；当已注册用户想查看自己购物篮中的物品，后台服务器就必须从数据库中找出该用户已挑选出的物品的详细信息，产生输出结果，反馈到用户浏览器中。

静态网站和动态网站各有特点，搭建网站采用动态还是静态技术主要取决于网站的功能需求和内容的多少，如果网站功能比较简单，内容更新量不是很大，采用静态网站的方式会更简单，反之一般要采用动态网站技术来实现。

静态网站可以使用 Frontpage 或 Dreamweaver 等网页编辑工具来建立，而动态网站需要使用服务器端网页技术，如本书介绍的 ASP.NET 来搭建。

1.2　ASP.NET 与 VWD 2008

ASP.NET 是由 Microsoft 公司提出的动态网站设计技术与程序框架，它带给人们的是全新的技术，和由此产生的开发效率的提高，网站性能的提升。使用 ASP.NET 提供的内置服务器控件或第三方生成的控件，可以创建既复杂又灵活的用户界面，大幅度减少生成动态网

页所需的代码，同时，ASP.NET 能够在服务器上动态编译和执行这些控件代码。

目前，ASP.NET 作为 Windows 平台上流行的网站开发工具，能够提供各种方便的 Web 开发模型，利用这些模型用户能够快速地开发出动态网站所需的各种复杂功能。

1.2.1　ASP.NET 的历史

早期的 Web 程序开发是十分繁琐的事情，一个简单的动态页面就需要编写大量的代码(一般用 C 语言)才能完成。

1996 年，Microsoft 推出了 ASP(Active Server Page)1.0 版。它允许采用 VBScript/JavaScript 这些简单的脚本语言编写代码，允许将代码直接嵌入 HTML，从而使得设计动态 Web 页面的工作变得简单。在进行程序设计时，ASP 能够通过内置的组件，实现强大的功能(如 Cookie)。ASP 最显著的贡献就是推出了 ActiveX Data Objects(ADO)，它使得程序对数据库的操作变得十分简单。

1998 年，微软发布了 ASP 2.0 和 IIS 4.0。与前版相比，2.0 版最大的改进是外部的组件需要初始化。用户能够利用 ASP 2.0 和 IIS 4.0 建立各种 ASP 应用，而且每个组件有了自己单独的内存空间，可以进行事务处理。

随后，微软公司开发了 Windows 2000 操作系统，其中的 Windows 2000 Server 系统提供了 IIS 5.0 和 ASP 3.0。此次升级，最主要的改变就是把很多事情交给 COM+来做，效率比以前的版本更高，而且更稳定。

ASP.NET 是 Microsoft 公司于 2002 年推出的新一代体系结构——Microsoft .NET 的一部分，用来在服务器端构建功能强大的 Web 应用，包括 Web 窗体(Web Form)和 Web 服务(Web Services)两部分。伴随着.NET 技术的出现，ASP.NET 1.0 也应运而生。ASP.NET 1.0 在结构上与前面的 ASP 版本截然不同，几乎完全是基于组件和模块化的。ASP.NET 1.0 允许开发者以一种非常灵活的方式创建 Web 应用程序，并把常用的代码封装到各种面向对象的组件中，这些组件可以由客户端用户通过事件来触发。同时，ASP.NET 提出了代码隐藏类(CodeBehind)的概念，把逻辑代码(.aspx.cs)和表现页面(.aspx)分离开来，使用户很容易使用后台代码来控制页面的逻辑功能。

2003 年，Microsoft 公司发布了 Visual Studio .NET 2003(简称 VS 2003)，提供了在 Windows 操作系统下开发各类基于.NET 框架的全新的应用程序开发平台。

2005 年，.NET 框架从 1.0 版升级到 2.0 版，Microsoft 公司发布了 Visual Studio.NET 2005(简称 VS 2005)。相应的 ASP.NET 1.0 也得到了升级，成为 ASP.NET 2.0。它修正了以前版本中的一些 Bug 并在移动应用程序开发，代码安全以及对 Oracle 数据库和 ODBC 的支持等方面都做了很多改进。

2008 年， Visual Studio.NET 2008(简称 VS 2008)问世了，ASP.NET 相应的从 2.0 版升级到 3.5 版。其中 ASP.NET 3.5 版最重要的功能提高在于：支持 AJAX 的网站，改进了对语言集成查询(LINQ) 的支持。这些改进提供了新的服务器控件和类型、新的面向对象的客户端类型库等功能。

1.2.2 ASP.NET 的开发环境

Visual Studio.NET 2008 集成了 Visual Basic 2008、Visual C# 2008、 Visual C++ 2008 和 Visual Web Developer 2008 等工具，是一个功能强大的集成开发平台，几乎可以在其中完成所有的软件开发任务：如可以在 VS.NET 2008 中进行 Windows 应用程序开发、ASP.NET Web 应用程序开发、XML Web Services 开发、设备应用程序开发以及与其相关的数据库应用处理。

Visual Studio.NET 2008 针对不同需求，提供了多个版本，分为面向大型软件开发团队的 Visual Studio Team System、面向一般开发者的 Professional、standard 和 Express 四个版本。其中的 Visual Studio.NET 2008 Express（速成版）是微软免费为初学者提供的。

进行 ASP.NET 开发时需要使用的语言是 Visual Basic.NET 或者是 C#，这两种语言都是 Visual Studio.NET 环境下的程序设计语言，但并不是必须使用 Visual Studio.NET 环境才能进行 ASP.NET Web 程序设计。因为 ASP.NET 文件实际上是一个纯文本的文件，编译工作是在用户向服务器第一次发出对该文件的 HTTP 请求时由 Web 服务器进行的，并不是由 VS 完成的。所以网页性能的好坏与代码设计环境是否是 VS 无关，甚至用户可以在文本编辑器（如写字板）中进行 ASP.NET 代码的设计。

在本书中，以 C#作为编程语言，以 VS 2008 Express 中的 Visual Web Developer 2008（简称 VWD 2008）组件作为开发环境，进行 ASP.NET 动态网页设计和网站开发。Visual Web Developer 2008 是一个功能齐备的开发环境，可以方便、高效地创建 ASP.NET Web 应用程序（简称网站）。

Visual Web Developer 2008 提供下列功能。

- 网页设计：VWD 2008 内置功能强大的网页编辑器，包含所见即所得的编辑模式和 HTML 编辑模式，以及智能感应功能和验证功能。支持所见即所得的拖拽界面，可以创建美观、易用的网站。
- 网页设计功能：VWD 2008 支持页面模式，使用主题和外观保持一致的页面外观，从而统一管理网页的排版与布局。
- 代码编辑：VWD 2008 提供代码编辑器，使用户可以使用 Visual Basic.NET 或 C#编写动态网页的代码。代码编辑器包括语法修饰和智能感应功能。
- 调试：提供调试器，帮助用户查找程序中的错误。
- 控件：ASP.NET Web 服务器控件整合了创建网站所需的大部分功能，用户可以快速开发 Web 应用程序。
- 数据访问：支持用户在网页中显示和编辑数据。数据可以位于各种数据存储区中，其中包括数据库或 XML 文件。在很多情况下，用户无需编写任何代码，即可向网页中添加和编辑数据。
- 安全性、个性化设置：提供内置的应用程序服务，用户可以向网站中添加用于确保登录安全性的成员资格；提供配置文件属性，维护用户特定的信息；另外还提供其他功能，其中的大部分功能都不要求编写任何代码。
- 对文件传输协议(FTP)的内置支持：使用 VWD 2008 的 FTP 功能，可以直接连接到服务器，然后在该服务器上创建和编辑文件。

此外，Visual Web Developer 2008 包含了一个内置的 Web 服务器，方便开发人员创建和调试 ASP.NET Web 应用程序。因此，用户不需要再安装和配置 IIS 服务器，就可以开发 ASP.NET Web 应用程序。

1.3　安装 VWD 2008 开发环境

下面，以 Visual Studio 2008 Express 为例，介绍 Visual Web Developer 2008(本书以后全部简称为 VWD 2008)的安装步骤。

(1) 将 Visual Studio 2008 速成版安装光盘置入光驱，稍等一下，系统自动启动安装程序，如果没有自动启动，请双击光盘中的 setup.exe 程序来启动安装向导，就可以看到安装欢迎界面，如图 1-3 所示。



图 1-3　安装欢迎界面

(2) 选择 Visual Web Developer 2008 选项，出现软件用户许可协议。

(3) 选中【我接受许可协议中的条款】复选框，同意授权，单击【下一步】按钮安装组件。

(4) 默认安装路径是 C:\Program Files\Microsoft Visual Studio 8，用户可以根据需要指定不同的安装路径。

(5) 单击【下一步】按钮开始复制组件，用户可以看到将要安装的组件。复制和安装完成后，出现完成安装界面。

(6) 单击【完成】按钮，完成 Visual Web Developer 2008 Express 的安装。

1.4　开发 ASP.NET Web 程序流程

开发 ASP.NET Web 应用程序，就是创建基于 ASP.NET 的 Web 网站。一个 Web 网站通

常在计算机中对应于一个虚拟目录，目录中包含页、控件、代码模块和服务，以及配置文件和各种资源。

利用 VWD 2008 开发 Web 应用程序，一般分为 3 大阶段：首先需要进行整体规划、利用模板创建 Web 构架，然后进行开发调试，最后进行网站的发布工作。

- 规划，创建动态网站：需要确定所制作的网站主题、风格、页面元素、逻辑结构等。
- 开发调试：这一阶段工作包括系统数据库的实现、基本类文件的实现、Web 页面实现等。
- 网站发布：发布的服务器可以是远程，也可以是本地，上传网站可利用 FTP 工具。

下面，通过具体的实例来说明如何在 VWD 2008 中创建 ASP.NET Web 应用程序。

1.4.1　使用 VWD 2008 创建 ASP.NET 网站

当第一次启动 VWD 2008 时，系统会自动进行相关的初始化环境设定。

打开【开始】|【程序】|【Visual Web Developer 2008 Express Edition】，启动 VWD 2008 速成版，出现起始页，如图 1-4 所示。

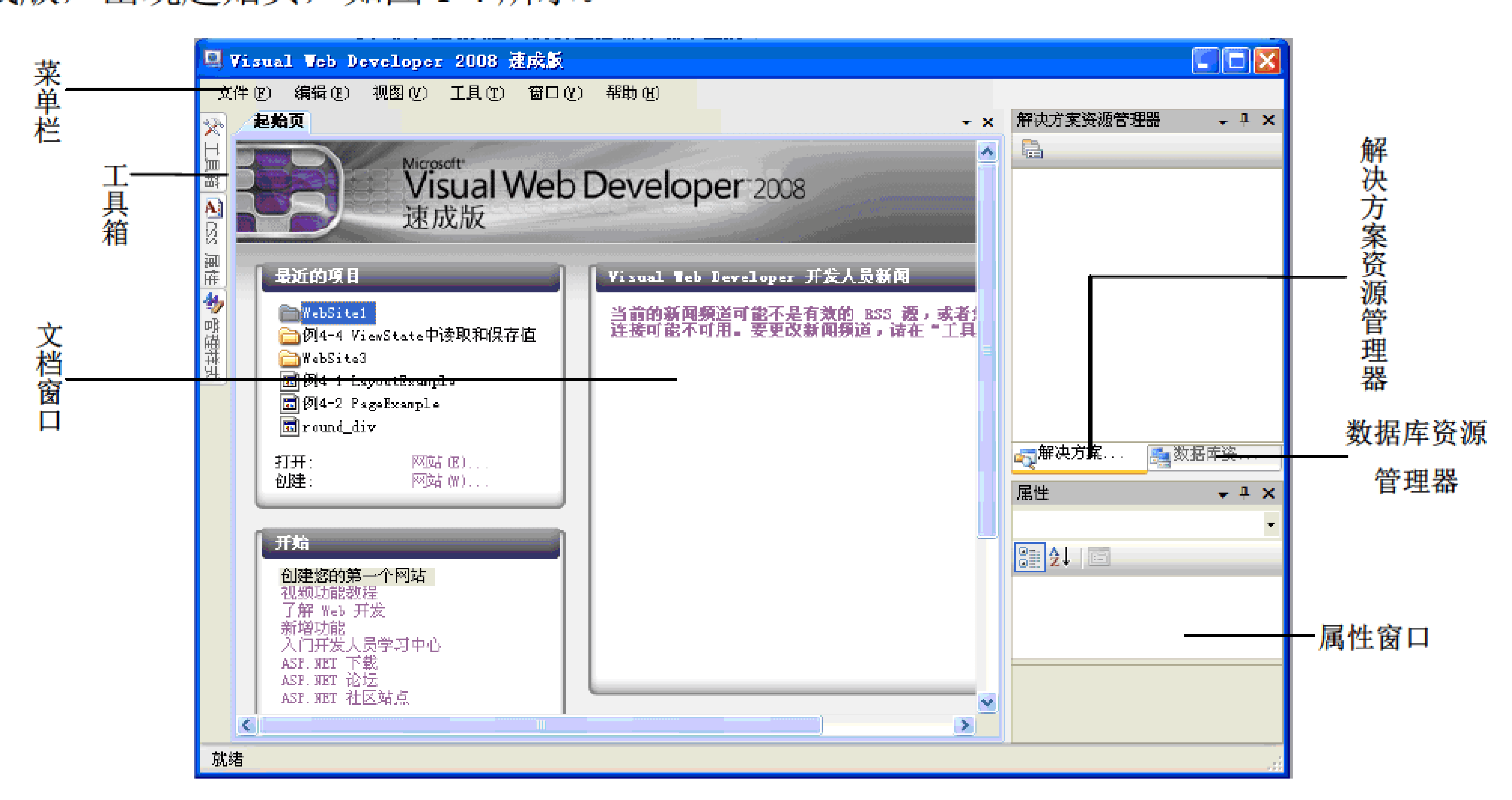


图 1-4　VWD 2008 开发环境示意图

在进行网站建设之前，先熟悉一下 VWD 2008 的开发环境。图 1-4 显示了在 Visual Web Developer 2008 中可用的窗口和工具。

- 菜单栏：提供格式化文本、查找文本等命令。一些工具只有在设计视图下才可用。
- 工具箱：提供可以拖到页面上的控件和 HTML 元素。工具箱元素按常用功能分组。
- 文档窗口：显示用户正在选项卡式窗口中处理的文档。单击选项卡可以实现在文档间切换。

文档窗口通常有两个视图：图形设计视图和源视图。

图形设计视图允许在用户界面或网页上指定控件和其他项的位置，可以从工具箱中拖动控件，并将控件置于设计图上。

源视图用于显示文件或文档的源代码，此视图支持编码帮助功能，如智能感应功能、可

折叠代码节、重构和代码段插入等。

另外，文档窗口还提供混合视图，通过该视图可以同时查看文件的图形和代码视图，此视图称为拆分视图。

- 解决方案资源管理器：显示网站中的文件和文件夹。通过“解决方案资源管理器”，可以快速打开文件进行编辑，向项目中添加新文件，以及查看方案、项目和项目属性等。
- 数据库资源管理器：显示数据库连接。
- 属性窗口：允许用户更改页、HTML 元素、控件和其他对象的设置。

【例 1-1】创建一个 ASP.NET 网站，存放至新建文件夹 D:\ myasp 中，并观察组成网站的基本文件。

(1) 在 D 盘新建文件夹 myasp，用于存放网站文件。

(2) 在 VWD 2008 菜单栏中选择【文件】命令，在其下拉子菜单中，可以看到 VWD 2008 支持创建网站或文件，如图 1-5 所示。

(3) 选择【新建网站】命令，出现【新建网站】对话框。在列出的【模板】中选择“ASP.NET 网站”，在【位置】下拉列表中选择“文件系统”，【路径】选择已建立的文件夹 D:\ myasp，在【语言】下拉列表中选择 visual c#，单击【确定】按钮，结束网站的创建，如图 1-6 所示。



图 1-5　VWD 2008 文件菜单

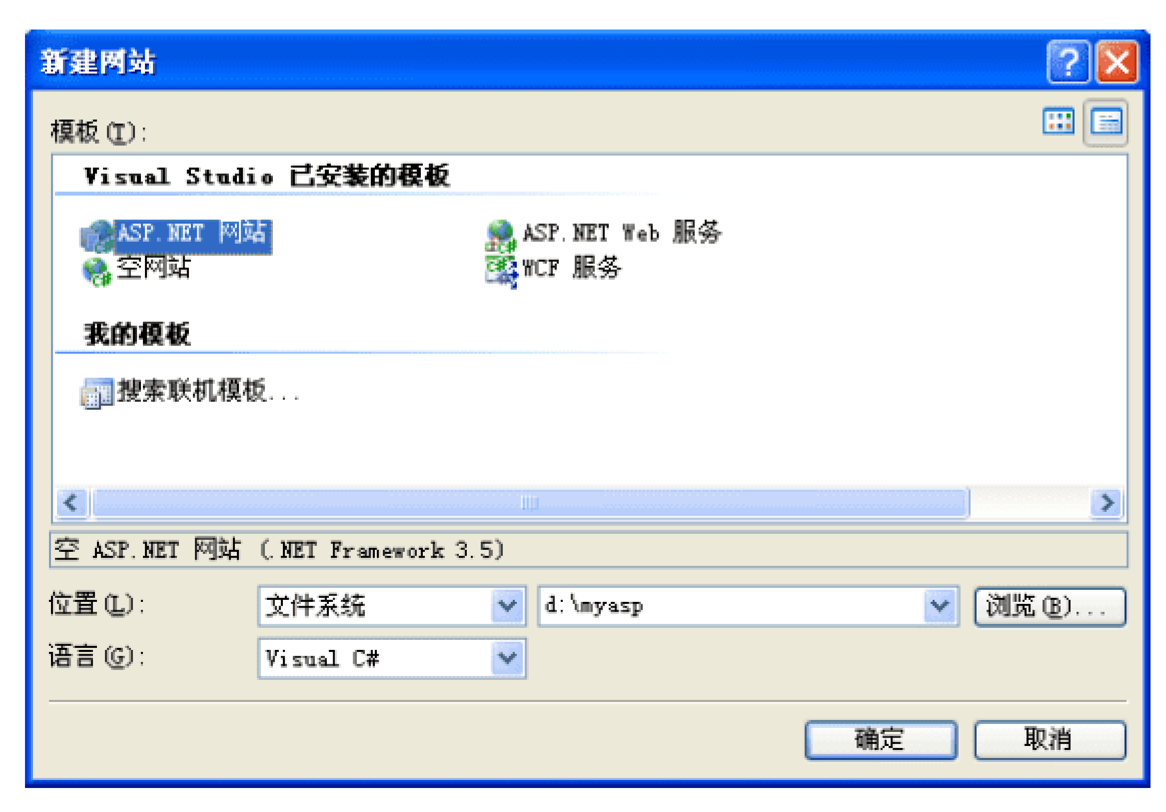


图 1-6　“新建网站”对话框

说明：

a. 创建网站时需要指定一个模板。每个模板创建包含不同文件和文件夹的 Web 应用程序。打开 D 盘 myasp 文件夹，可以看到，一个 ASP.NET Web 网站包含名为 App_Data 的子文件夹和 3 个初始文件，如图 1-7 所示。

- Default.aspx 文件为 ASP.NET Web 窗体文件，该文件作为网站的默认主页，可以包含 Web 控件。以 .aspx 作为页面扩展名的页面由 ASP.NET 运行，而扩展名为 .htm 和 .html 的网页是不被 ASP.NET 运行的。
- Default.aspx.cs 文件为网页运行时要编译的类源代码文件。

● web.config 文件为 Web 配置文件。

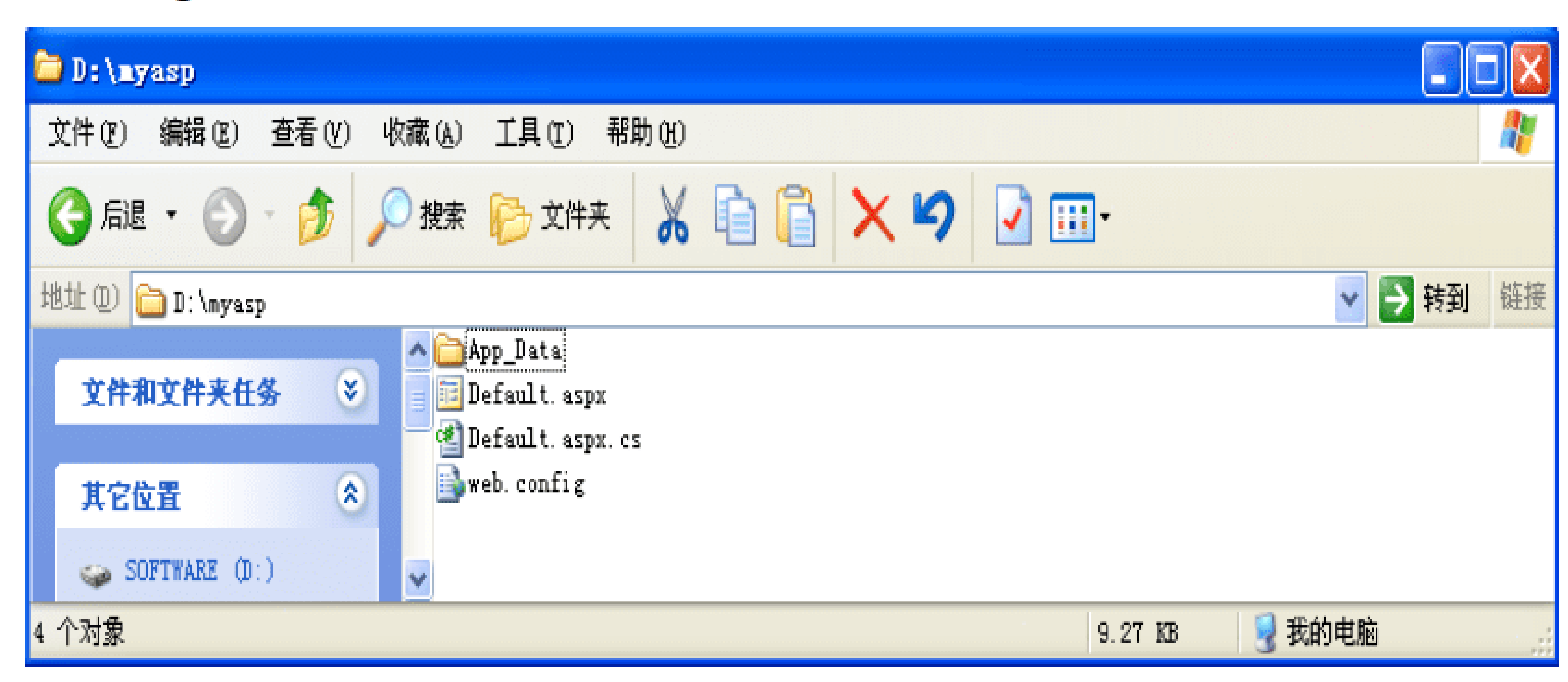


图 1-7　组成动态网站的基本文件

b. 在【位置】下拉列表中，有以下几项选项。

● 文件系统网站，指用户选择在本地计算机上某个文件夹中存储页面和其他文件。

● HTTP 网站，将用户的文件存储在本地 IIS 根目录(通常是 \Inetpub\Wwwroot\)的子文件夹中。

● FTP 网站，将文件存储在远程服务器上，用户可以使用文件传输协议(FTP)通过 Internet 访问该服务器。

c. 用户选择的编程语言将是网站的默认语言。但在同一个 Web 应用程序中可以使用多种编程语言创建页面和组件。

(4) 观察代码。创建网站结束，VWD 2008 自动打开 Default .aspx 文件，默认情况下以“源”视图显示该页，在该视图下可以查看页面的代码，如图 1-8 所示

此文件是由 XHTML 语言编写的，关于 XHTML 语言的格式及语法规则，将在第 2 章详细讨论。

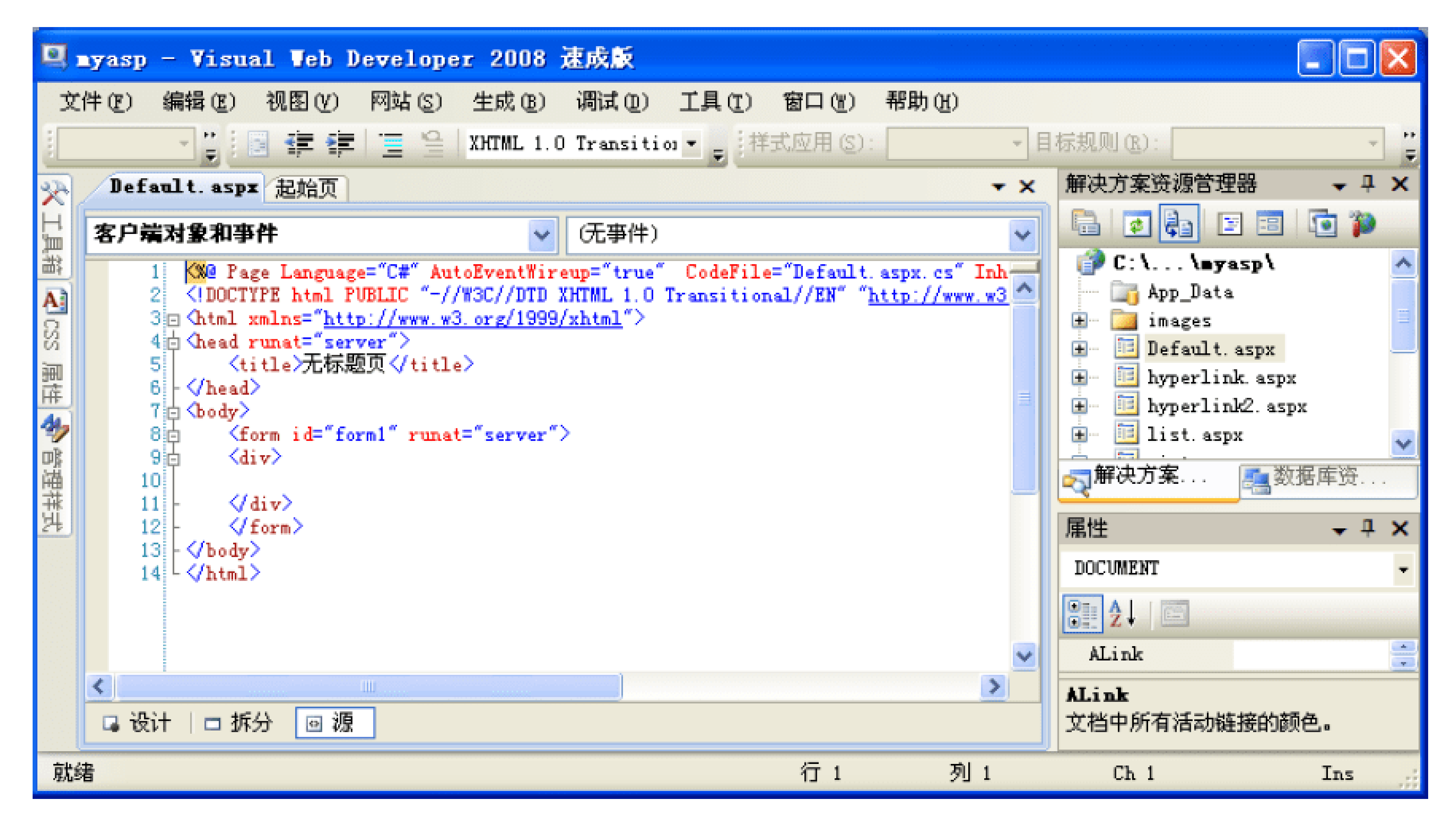


图 1-8　在源视图中查看 Default .aspx 代码

1.4.2　新建 ASP.NET 页面

建立了 Web 网站，就可以在网站中新建或打开 ASP.NET 页面。

【例 1-2】在刚才创建的网站下，创建一个 ASP.NET 页面，存储为 welcome.aspx。

(1) 在 VWD 2008 菜单栏中选择【文件】|【新建文件】命令，出现【添加新项】对话框，如图 1-9 所示。

(2) 在列出的【模板】中选择“Web 窗体”，在【名称】中输入创建的页面名称，本例名称为 welcome.aspx，在【语言】下拉列表中选择 visual c#，选中【将代码放在单独的文件中】复选框。单击【添加】按钮完成 welcome.aspx 页面的创建。



图 1-9　创建 welcome.aspx

说明：

添加页面的另一种方法是在【解决方案资源管理器】中右键单击网站名称，在出现的快捷菜单中选择【添加新项】。

(3) VWD 2008 创建新页面并以源视图打开。可以在这里看到相应的代码。

(4) 向页中添加一些文本。在文档窗口的底部，单击【设计】选项卡切换到设计视图。设计视图以所见即所得的方式显示用户正在使用的页。新建立的页面没有任何文本或控件，因此页面是空白的。在页面上键入“欢迎使用 Visual Web Developer”，同时在设计视图中就看到了键入的文本，如图 1-10 所示。这时，切换到源视图，可以看到通过在设计视图中键入内容而创建的源文件代码，如图 1-11 所示。

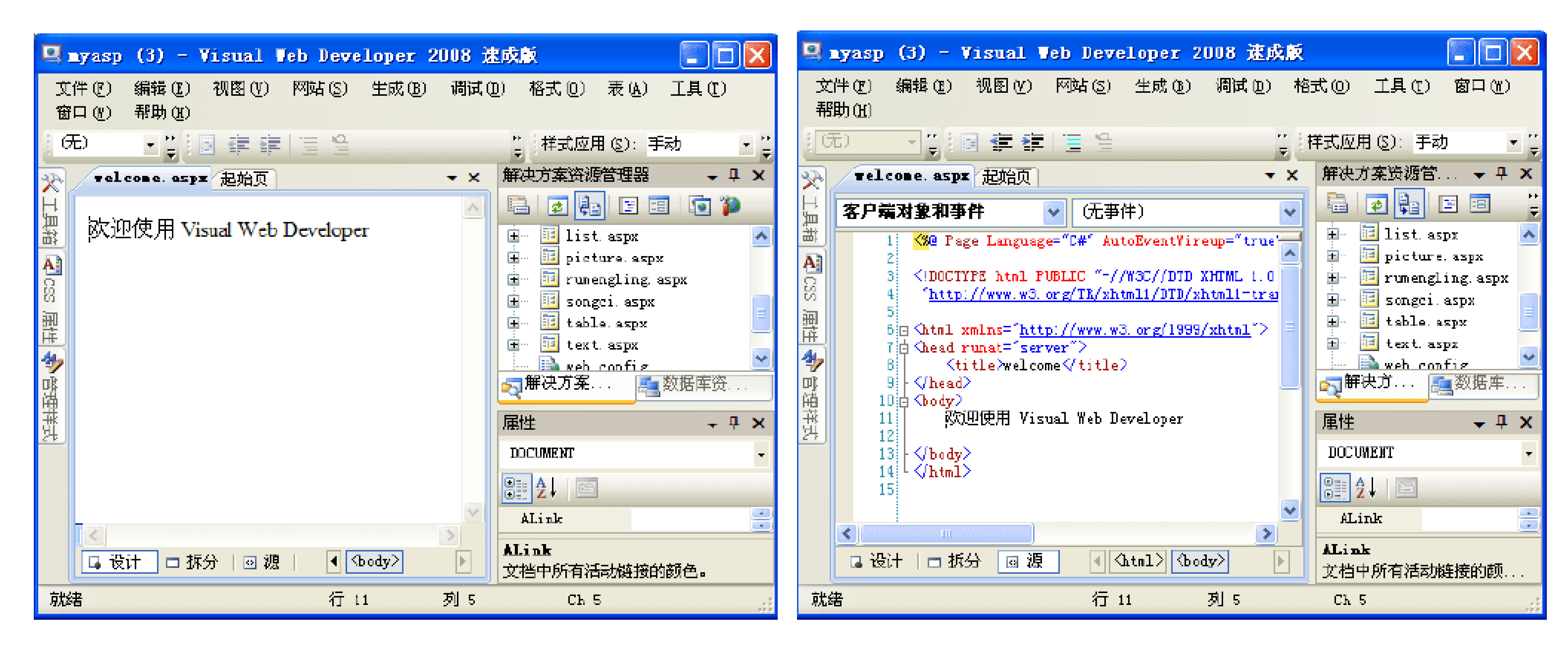


图 1-10　在设计视图中查看 welcome.aspx　　　　图 1-11　在源视图中查看 welcome.aspx

1.4.3　在 VWD 2008 中执行页面

在继续向页面添加控件，进入深入学习之前，读者可以执行该页，测试显示结果。要运行页，需要一个 Web 服务器，如果在发布的网站中，执行页面必须使用 IIS 作为 Web 服务器，但对于这里建立的文件系统网站而言，VWD 2008 中的默认 Web 服务器 ASP.NET Development Server，使系统在没有安装 IIS 服务器时，也可以测试 ASP.NET 程序的执行结果。

【例 1-3】在 VWD 2008 中执行 welcome.aspx。

(1) 在 VWD 2008 的菜单栏中选择【文件】|【打开网站】命令，出现【添加新项】对话框，选择网站及文件 welcome.aspx。

(2) 在菜单栏中选择【文件】|【在浏览器中查看】命令，网页显示在浏览器中，如图 1-12 所示。

说明：

运行网站的快捷方法是使用组合键 Ctrl+F5。



图 1-12　welcome.aspx 运行结果

现在创建的网站内容还比较简单，以后的学习将会逐渐丰富该网站。

实验 1　ASP.NET 与 VWD 2008 开发平台

1. 实验目的

安装 ASP.NET 的开发环境 VWD 2008，并创建简单的动态网站。

2. 实验内容和要求

(1) 安装 VWD 2008，建立 ASP.NET 3.5 的运行环境。

(2) 使用 VWD 2008 创建网站 lianxi1，并创建第一个页面 exam1.aspx。

(3) 单击页面 exam1.aspx 的设计标签，切换到页面设计窗口，在页面中输入：“这是我的第一个 ASP.NET 程序”。

(4) 运行这个页面。

第2章　XHTML基础知识

HTML 是互联网上组建网站的最常用语言，但由于它本身固有的缺陷，已经不能满足动态网站的组建和互联网发展的需求。XML 是“万维网联盟”定义的受到广泛支持的行业标准，是新一代互联网的关键，但 XML 不兼容 HTML，因此在现实中用 XML 取代 HTML 还有很长的路要走。XHTML 是基于 XML 的规范，并对 HTML 进行增强而形成的一种过渡语言。

本章对 XHTML 的概念、页面结构、语法规则和标记进行了详细的描述。通过本章的学习，读者能够掌握 XHTML 的基本概念，并会使用 XHTML 编写 ASP.NET 网页。

本章的学习目标

- 理解什么是 HTML、XML 和 XHTML，以及三者之间的关系
- 掌握动态网页的组成结构
- 掌握 XHTML 的语法规则
- 熟悉并使用 XHTML 标记

2.1　Web 基本技术

互联网技术正处于日新月异的高速发展中，它汇集了当前信息处理的几乎所有技术手段，来满足用户的需求。在这里对 Web 基本技术进行讨论。

2.1.1　HTML

HTML(HyperText Markup Language)超文本标记语言，是制作页面文档的主要编辑语言。无论在何种操作系统下，只要有浏览器就可以运行 HTML 页面文档。作为一种标记语言，HTML 利用近 120 种标记来标识网页的结构及超链接等信息，使页面在浏览器中展示出精彩纷呈的效果。HTML 只是建议 Web 浏览器应该如何显示和排列信息，并不能精确定义格式，因此在不同的浏览器中显示的 HTML 文件效果会不同。

HTML 文件是一种纯文本文件，通常以.htm 或.html 作为文件扩展名。可以用各种类型的工具来创建或者处理 HTML 页面，如记事本、写字板、FrontPage、Dreamweaver 等。

HTML 由于它的简单易学，得到了广泛的使用，但是 HTML 存在着不可克服的缺陷。

首先，HTML 的标记是固定的。也就是说，HTML 不允许用户创建自己的标记。所以

HTML 很难做更复杂的事情：如它无法描述矢量图形、科技符号和一些其他的特殊显示效果。

其次，HTML 中标记的作用只是建议浏览器用何种方式显示数据。HTML 语言无法解释数据之间的关系，以及相关结构方面的信息，因此不能适应日益增多的信息检索要求和存档要求。

通过上面的讨论可以看出，HTML 尽管很简单方便，但当需要对一定量的数据进行复杂处理时，就力不从心了，而这正是 XML 可以大显身手的地方。

2.1.2　XML

HTML 是很成功的标记语言，目前很多网站是由 HTML 语言制作的。HTML 语法要求比较松散，这对网页编写者来说，比较方便，但对计算机来说，语言的语法越松散，处理起来就越困难，传统的计算机能够处理松散的语法，但随着互联网的发展，对于许多新兴的连接到互联网的设备，比如手机，解析网页语法的难度就比较大。于是，人们开始致力于构建另一个标记语言，使它既具有 HTML 的简单性，又具有强大的功能和可扩展性，XML 应运而生。

XML(eXtensible Markup language)可扩展标记语言，将网络上的文档规范化，并赋予标记一定的含义，同时，XML 又不仅仅只是标记语言，它还提供了一个标准，用户可以利用这个标准定义新的标记语言，并为这个新的标记语言规定它所特有的一套标记。

XML 已经在文件配置、数据存储、基于 Web 的 B2B 交易、存储矢量图形和描述分子结构等众多方面得到广泛的应用。但是，由于目前的浏览器对 XML 的支持还不够完善，XML 在互联网上完全替代 HTML 还需要很长一段时间。

在由 HTML 向 XML 过渡阶段，国际万维网组织(W3C)在 HTML 基础上，按照 XML 格式制定了新的规范 XHTML 1.0，使网络编程人员只要通过简单的更改，就能将 HTML 转为 XHTML，从而为实现由 HTML 向 XML 的过渡找到桥梁。

2.1.3　XHTML

XML 虽然数据转换能力强大，完全可以替代 HTML，但面对成千上万已有的基于 HTML 语言设计的网站，直接采用 XML 还为时过早。因此，在 HTML 4.0 的基础上，用 XML 的规则对其进行扩展，得到了 XHTML(eXtensible Hypertext Markup Language)可扩展超文本标记语言。XHTML 是为了使 HTML 向 XML 顺利过渡而定义的标记语言，它以 HTML 为基础，采用 XML 严谨的语法结构，可以说，XHTML 结合了部分 XML 的强大功能及大多数 HTML 的简单特性，是一种增强了的 HTML，它的可扩展性和灵活性将适应未来网络应用的需求。越来越多的程序员开始利用 XHTML 设计网站结构，编写网页内容。

目前国际上在网站设计中推崇的 Web 标准就是基于 XHTML 的应用(即通常所说的 CSS＋DIV)。本书第 1 章所介绍的 VWD 2008 开发工具默认建立的网页即为 XHTML 格式的网页。

大部分的浏览器都可以正确地解析 XHTML，即使老版本的浏览器，也将 XHTML 作为 HTML 的一个子集。因此，可以说几乎所有的网页浏览器在正确解析 HTML 的同时，都可以兼容 XHTML。

2.2　XHTML 的基本格式

XHTML 以 HTML 为基础，因此与 HTML 有很多相似之处。接下来介绍 ASP.NET 的页面文档结构和 XHTML 的语法规则。

2.2.1　ASP.NET 的文档结构

这里以第 1 章建立的 welcome.aspx 为例，来说明 ASP.NET 的文档结构。

welcome.aspx 的 XHTML 代码：

```
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="welcome.aspx.cs"
Inherits="welcome" %>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <script runat="server">
    </script>
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>无标题页</title>
    </head>
    <body>
        <form id="form1" runat="server">
        <div>
          <p>
            欢迎使用 Visual Web Develope
               
         </p>
        </div>
        </form>
     </body>
    </html>
```

从上面的代码可以看到，一个完整的 ASP.NET 页面文档是由指令、文档类型声明、代码声明、服务器代码、文本和 XHTML 标记等部分组成。

(1) 指令

ASP.NET 页面通常包含一些指令，允许用户指定页面的属性和配置信息，对页面进行设置。指令指定的设置，不会出现在浏览器端。

在网页设计时，ASP.NET 提供“代码分离”技术，使开发者进行分工协作，分别进行网页界面代码设计和后台服务器运行代码设计。具体在实践中，将源代码放在扩展名为.aspx 文件中，将 Web 服务器运行代码放在另一个文件中，若此文件是由 C#编写的，则文件扩展名为.cs。这样做可以使前台 HTML 界面随着潮流不停的变化，而后台服务器端的代码可以稳定地实现业务处理。

.aspx 文件和.cs 文件的相互关联是由 aspx 文件中@page 指令连接的。如本例中：

```
<%@ Page Language="C#"  AutoEventWireup="true"  CodeFile="welcome.aspx.cs"
Inherits="welcome" %>
```

该指令说明编程语言为 C#，需要链接的 cs 文档为 welcome.aspx.cs。

(2) 文档类型声明

DOCTYPE 为文档类型声明，指定本文档遵从的 DTD(Document Type Definition 文档类型定义)标准，同时指定了文档中的 XHTML 版本，可以和哪些验证工具一起使用等信息，以保证此文档与 Web 标准的一致。

文档类型声明是每个网页文档必需的，如果网页文档中没有文档类型声明，浏览器就会采用默认的方式，即 W3C 推荐的 HTML 4.0 来处理此 HTML 文档。

本例的文档类型声明部分为：

```
<！DOCTYPE  html  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">。
```

DOCTYPE 是 document type (文档类型)的缩写，W3C//DTD XHTML 1.0 Transitional 说明此文档符合 W3C 制定的 XHTML 1.0 规范，即声明此文档应该按照 XML 文档规范来配对所有标记。xhtml1-transitional.dtd 中的 DTD 是文档类型定义，包含了文档的规则，浏览器根据页面所定义的 DTD 来解释页面内的标识，并将其显示出来。

(3) 代码声明

包含 ASP.NET 页面的所有应用逻辑和全局变量声明、子例程和函数。页面的代码声明位于 <script>…</script>标记中。

(4) 服务器代码

大多数 ASP.NET 页面包含处理页面时在服务器上运行的代码。页面的代码位于 script 标记中，该标记中的开始标记包含 runat="server" 属性。

如本例中的<script runat="server">，说明页面运行时，ASP.NET 将此标记标识为服务器控件，并使其可用于服务器代码。

(5) 文本和 XHTML 标记

页面的文本部分用 XHTML 标记来实现，这一部分结构应完全符合 HTML 的文件结构。在上面的例子中可以看到，一个最基本的 HTML 网页结构由 3 个部分构成：

```
<html>
        <head>
                <title>标题内容</title>
        </head>
        <body>
              主要内容
        </body>
</html>
```

● <html >…</html>: 整个 HTML 文件的起止标记。

其他 HTML 标记都要被放在这对标记之间。

在 HTML 代码中，仅有<html >…</html>，而在 XHTML 代码中使用了<html xmlns="http://www.w3.org/1999/xhtml">…</html>。其中的 xmlns 是 XHTML namespace 的缩写，即 XHTML 命名空间，用来声明网页内所用到的标记是属于哪个名称空间的。本例中，指定 HTML 的标记名称空间为 http://www.w3.org/1999/xhtml，这属于 XML 1.0 的写法。说明整个网页标记应符合 XHTML 规范。

● <head>…</head>: HTML 头部文件。

头部文件中包含页面传递给浏览器的信息，这些信息作为一个单独的部分，不是网页的主体内容，但有时对于浏览器而言是很有用的。在头部文件中可以设置页面的标题、关键字、外部链接和脚本语言等内容：如用<title>…</title>标记来设置网页的标题，用<script >…</script>标记来插入脚本等。

● <body>…</body>: 文档内容部分。

<body>…</body>标记之间为页面文档的主体，用来放置页面的内容，是在浏览器中需要显示的内容。对一个最简单的网页来说，<body>…</body>标记符是必须使用的标记符。

2.2.2 XHTML 的语法规则

因为引入 XHTML 的目的是在 HTML 中使用 XML，所以 XHTML 的语法规则比 HTML 严格很多。具体有下列规则。

(1) UTF-8 之外的编码，文档必须具有 XML 声明

当文档的字符编码是默认的 UTF-8 之外的编码时，编程人员必须在 XHTML 页面中添加一个 XML 声明并指定代码。

例如：<? xml version="1.0" encoding="iso-8859-1"?>

(2) 页面中的 HTML 标记之前必须使用 DOCTYPE 声明

XHTML 1.0 提供了 3 种 DTD 声明供选择，DOCTYPE 声明必须引用其中一种类型。

● Transitional(过渡型)：要求使用宽松的 DTD，这时可以使用符合 HTML 4.0 标准的标记，但是必须符合 XHTML 的语法。这是 ASP.NET 所采用的默认文档类型定义。

声明代码为：<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

● Strict(严格型)：要求使用严格的 DTD，这时不能使用任何表现层的标记和属性，如标记
。

声明代码为<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

● Frameset(框架型)：专门针对框架页面设计使用的 DTD，如果在网页中包含有框架，则需要采用这种 DTD。

声明代码为<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

(3) 页面的 html 标记必须指定命名空间

html 标记必须指定 XHTML 命名空间，即将 namespace 属性添加到 html 标记中。如例子中的<html xmlns="http://www.w3.org/1999/xhtml">…</html>。

(4) 文档必须包含完整的结构标记

文档必须包含 head，title 和 body 结构标记。框架集文档必须包含 head，title 和 frameset 结构标记。

(5) 标记必须正确嵌套

XHTML 要求有严谨的结构，文档中的所有标记必须按顺序正确嵌套，例如：

<p>This is a <i> bad example.</p></i>是错误的。

<p>This is a <i> good example.</i></p>是正确的。

也就是说，一层一层的嵌套必须是严格对称。

(6) 标记必须成对使用，若是单独不成对的标记，在标记最后加/>结束。如：

是错误的。

是正确的。

(7) 所有标记名称和属性的名字都必须使用小写

与 HTML 不同，XHTML 对大小写是敏感的，XHTML 要求所有的标记和属性的名字都必须使用小写。<title>和<TITLE>在 XHTML 是不同的标记。

(8) 属性值必须用引号""括起来

在 HTML 中，不要求给属性值加引号，但是在 XHTML 中，属性值必须被加引号。例如<height=80> 必须修改为：<height="80">。

特殊情况下，若用户需要在属性值里使用双引号，可以使用'表示，例如：

```
<alt="say&apos;hello&apos;">
```

(9) 属性不允许简写，每个属性必须赋值

XHTML 规定所有属性都必须有一个值，没有值的就重复本身。例如：

```
<input type="checkbox" name="shirt" value="medium" checked>
```

必须修改为：

```
<input type="checkbox" name="shirt" value="medium" checked="checked">
```

(10) 使用 id 替代 name 属性

(11) 图片必须有说明文字

每个图片标记必须有 ALT 说明文字。即必须对 img 和 area 标记应用文字说明 alt="说明"属性。

如：

(12) 不要在注释内容中使用--

--只能发生在 XHTML 注释的开头和结束，也就是说，在内容中它们不再有效。例如下面的代码是无效的： <!--这里是注释-----------这里是注释-->

可以用等号或者空格替换内部的虚线，如：<!--这里是注释====这里是注释--> 是正确的。

以上规则的使用是为了使代码有一个统一、唯一的标准，便于以后的数据再利用。为由 HTML 向 XML 过渡打下基础。

2.3　XHTML 标记、标记属性

标记(Tags)是指定界符(一对尖括号)和定界符括起来的文本，用来控制数据在网页中的编排方式，告诉应用程序(例如浏览器)以何种格式表现标记之间的文字。当需要对网页某处内容的格式进行编排时，只要把相应的标记放置在该内容之前，浏览器就会以标记定义的方式显示网页的内容。学习 XHTML 语言的重点就是学习标记的使用。

标记控制文字显示的语法为：

```
<标记名称>
    需进行格式控制的文字
</标记名称>
```

在 XHTML 标记中，往往还可以通过设定一些属性，来描述标记的外观和行为方式，以及内在表现，以便对文字编排进行更细微的控制。几乎所有的标记都有自己的属性。例如 style="text-align:center"，其中，style 就是标记的属性，style 的值设置文本格式为居中对齐。

使用标记符有如下一些注意事项：

- 任何标记都用<和>括起来，一般情况下，标记是成对出现的。
- 标记名与<之间不能有空格。
- 某些标记要加上属性，而属性只能加于起始标记中。格式为：

```
<标记名　属性名=属性值　 属性名=属性值 …> 网页内容 </标记名>
```

XHTML 文件支持很多种标记，不同的标记代表不同的含义。XHTML 常用的标记包括主体标记、注释标记、分层标记、文本标记、列表标记、表格标记、图像标记和超链接标记等。

2.3.1　主体标记<body>…</body>

主体标记之间定义了网页的所有的显示内容。网页默认的显示格式为：白色背景，12 像素黑色 Times New Roman 字体。

在 XHTML 中，<body>标记用属性 style 来设置样式，如设置字体的大小、颜色、页面的背景色和背景图等。格式为：

```
<标记 style="样式 1：值 1；样式 2：值 2；……">
```

其中，样式与值用冒号分隔，如果 style 属性中包含多个样式，各个样式之间用分号隔开。style 属性常用的样式有以下几种。

- background-color：设置网页的背景颜色，默认为白色背景。
- color：设置网页中字体的颜色，默认颜色为黑色。
- font-family：设置网页中字体的名称，如宋体、楷体、黑体等。
- font-size：设置网页中字体的大小。
- text-align：设置网页中文本的对齐方式，常用有 3 种不同的取值：left(左对齐，默认对齐方式)、right(右对齐)、center(居中对齐)。

例如：<body style="font-family:宋体;color:blue">，设置网页字体为宋体，字体的颜色为蓝色。

2.3.2　注释标记<!--注释内容-- >

浏览器会自动忽略注释标记中的文字(可以是单行也可以是多行)而不显示。注释标记常用在比较复杂或多人合作设计的页面中，为代码部分加上说明，方便日后修改，增加页面的可读性和可维护性。

2.3.3　分层标记<div>…</div>

分层标记用来排版大块的 XHTML 段落，为 XHTML 页面内大块(block-level)的内容提供结构和背景的标记。可用 style 属性，在其中加入许多其他样式，以实现对其中包含元素的版面设置。

div 标记除了可以作为文本编辑功能外，还可以用作容器标记，即将按钮、图片、文本框等各种标记放在 div 里面作为它的子对象元素处理。

2.3.4　文本和格式标记

网页中最常用的就是文字了，下面将详细讲解 XHTML 中对网页中的文字进行格式设计和排版的常用标记。

(1) 标题字体大小标记 <hn>…</hn>

设定网页的标题格式。由大至小，有 6 种设置标题格式的标记：<h1>、<h2>、<h3>、<h4>、<h5>和<h6>。

(2) 字体的加粗、斜体和下划线标记

- …标记：以加粗字的形式输出文本。
- <i>…</i>标记：以斜体字的形式输出文本。
- <u>…</u>标记：以下划线形式输出文本。

(3) 段落标记

- <p>…</p>

段落标记<p>…</p>的作用是将标记之间的文本内容自动组成一个完整的段落。

- 预格式化标记<pre>…</pre>

预格式化标记<pre>…</pre>使标记之间的文本信息能够在浏览器中按照原格式毫无变化的输出。它可以使浏览器中显示的内容与代码中输入的文本信息格式完全一样。

(4) 换行标记

<br / >用于添加一个回车换行，该标记没有结束标记，故在 XHTML 中以</>结束。在编写 XHTML 时，如果在文件中用回车键分开了某一段文字，当在浏览器中显示时，浏览器会忽略源代码中的换行，而并不会显示换行的效果。若要显示网页中的文字换行效果，必须在文件中使用
标记。

(5) 画线标记<hr />

画线标记< hr />单独使用，可以实现段落的换行，并绘制一条水平直线，并在直线的上下两端留出一定的空间。可以使用 style 属性进行设置。其中：

width 用于设置画线的长度，取值可以是以像素为单位的具体数值，也可以使用相对于其父标记宽度的百分比数值。

Height 用于设置画线的粗细，单位是像素。

(6) 文本居中标记<center>… </center>

文本居中标记用来将网页中 center 标记内的元素居中显示。

【例 2-1】建立 ASP.NET 页面，命名为text.aspx。

主体部分代码如下：

```
<body style="text-align:center;font-family:楷体_GB2312;color:blue">
        <!--设置整个页面的字体居中显示，字体为楷体，颜色为蓝色-->
    <form id="form1" runat="server">
    <div >
            设定标题格式示例：
          <h1>设定标题格式，此处用 h1 效果</h1>
          <h6>设定标题格式，此处用 h6 效果</h6>
                <hr style ="width:70%;height:10px;color:Black" />
                <!--画一条分割线，宽度为整个页面的 70%，宽度为 10 像素，颜色为黑色-->
          <p> 字体的特殊效果示例：</p>
          <b>粗体显示</b><br />
          <i>斜体显示</i><br />
                <hr />
      </div>
    </form>
</body>
</html>
```

在浏览器中查看，运行结果如图 2-1 所示。

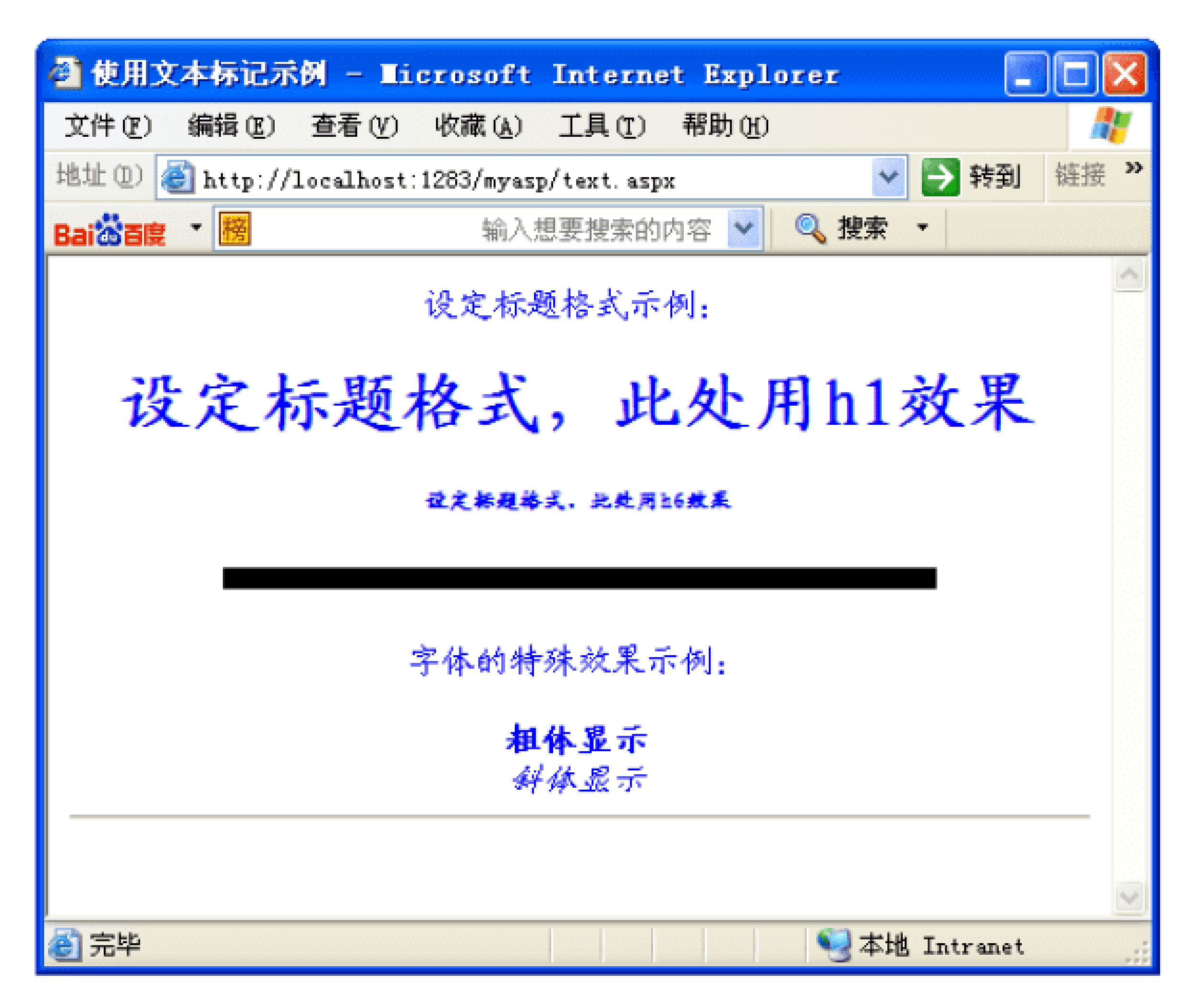


图 2-1　对页面使用文本标记

(7) 列表标记

使用列表标记为网页中的文本设置表格。包括有序列表标记和无序列表标记。

● 无序列表标记
…
和列表项标记
…

无序列表是指各个列表项目没有顺序，显示时，在各列表项前面显示特殊符号的缩排列表，语法格式为：

```
<ul  style="list-style-type">
<li>列表项 1
<li>列表项 2
…
<li>列表项 n
</ul>
```

其中，list-style-type 可以有几种形式：默认形式 disc(实心圆)●、circle(空心圆)和 square(实心方块)。

有自动换行的作用，每个条目自动为一行。每一个<li>创建的项目可以使用 list-style-type 单独指定的项目符号。

● 有序列表标记
…
和列表项标记
…

有序列表是在各列表项前面显示数字或字母的缩排列表，有序列表显示时，会在每个条目前面加上一定形式的有规律的项目序号。语法格式为：

```
<ol  style="list-style-type">
<li>列表项 1
<li>列表项 2
…
<li>列表项 n
</ol>
```

其中，list-style-type 可以设为：upper-alpha(大写英文)、lower-alpha(小写英文)、

upper-roman(大写罗马数字)、lower-roman(小写罗马数字)和 decimal(十进制数字)等。默认的列表标识符为阿拉伯十进制数字。

同无序列表中一样，<li>有自动换行功能。而且每一个<li>创建的项目可以使用 list-style-type 单独指定的项目符号。

【例 2-2】建立 ASP.NET 页面，名称为list.aspx。

其主体部分 XHTML 代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
         电子产品
       <ul>
            <li>数码相机</li>
            <li style ="list-style-type:disc">移动硬盘</li>
            <li style ="list-style-type:circle">MP3,MP4</li>
            <li style ="list-style-type:square">笔记本电脑</li>
       </ul>
         服装箱包
       <ol >
            <li>针织衫</li>
            <li style ="list-style-type:lower-roman">女鞋</li>
            <li style ="list-style-type:lower-alpha">男夹克</li>
            <li style ="list-style-type:upper-roman">流行男女箱包</li>
       </ol>
    </div>
    </form>
</body>
```

在浏览器中运行，结果如图 2-2 所示。



图 2-2　对页面使用列表标记

(8) 空格标记

在 XHTML 中，直接输入多个空格，仅仅会被视为一个空格，而多个回车换行符也仅仅被浏览器解读为一个空格。为了能够显示多个空格，XHTML 保留了 HTML 中的空格标记 。一个 代表一个空格；多个 则代表相应的空格数。

2.3.5 表格标记

通过学习上面的文本标记，已可以对网页内容设置字体、段落、对齐方式等，但是由于浏览器的不同，并不能精确控制文本具体显示在网页的位置，而使用表格标记就可以对网页中的各个元素的具体位置进行控制。因此，表格在网页设计中的定位功能是极其重要，同时也是在网页设计所有编辑方式中最灵活的。

表格由行与列组成，每一个基本表格单位称为单元格。单元格在表格中可以包含文本、图像、表单以及其他页面元素。

- 表格标记<table>… </table>

表格标记用来声明表格，标志着一个表格的开始和结束，表格的所有定义都在这对标记范围内适用。<table>… </table>标记的常用属性有以下几种。

  - align：设置表格在网页中的水平对齐方式，可选值 left、right、center。
  - backGround：为表格指定背景图片。
  - bgcolor：为表格设定背景色。
  - border：设置表格边框厚度，如果此参数为 0，那么表格不显示边界。
  - cellpadding：设置单元格中的数据与表格边线之间的间距，以像素为单位。
  - cellspacing：设置各单元格之间的间距，以像素为单位。
  - valign：设置表格在网页中的垂直对齐方式，可选值 top、middle、bottom。
  - width：设置整个表格宽度。

- 行起止标记<tr>… </tr>

此标记表明了表格一行的开始和结束，有以下属性。

  - align：设置行中文本在单元格中的水平对齐方式，可选值 left、right、center。
  - backGround：为这一行单元格指定背景图片。
  - bgcolor：为这一行单元格设定背景色。

- 单元格起始标记<td>… </td>

单元格起始标记用于设置表行中某个单元格的开始和结束。

【例 2-3】建立 ASP.NET 页面，名称为 table.aspx。

其 XHTML 主体部分代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
      <table border="5">
        <tr align ="center" >
        <td bgcolor="red"> 第一行第一列，背景红色 </td>
```

```
            <td bgcolor="blue">第一行第二列，背景蓝色 </td>
            <td bgcolor="green">第一行第三列，背景绿色 </td>
            </tr>
                <!--以上设置第一行，文字居中-->
             <tr >
             <td align="left"> 第二行第一列，左对齐 </td>
             <td align="center">第二行第二列，居中 </td>
             <td align="right">第二行第三列，右对齐 </td>
             </tr>
            <!--以上设置第二行，文字居左-->
          </table>
        </div>
        </form>
    </body>
```

在浏览器中查看运行结果，如图 2-3 所示。



图 2-3　对页面使用表格标记

2.3.6　超链接标记<a>…

超链接是通过文字、图像等载体对文件进行链接，引导文件的阅读。互联网的魅力就在于可以通过超链接使任何一个网页，可以任意链接到世界任何角落的其他网页文件。超链接往往用不同的颜色或下划线与网页中的其他文字区别，在阅读文件时，用户通过单击超链接，能够随时查阅文件相关的详细信息。

1. 超链接命令的格式

<a　href="URL"　id="设置锚点"　target="链接目标网页打开的窗口"> 锚点

锚点：实现链接的源点，通常当鼠标移动到锚标上会变成小手的形状，浏览者通过在锚标上单击鼠标就可以到达链接目标点。

href 属性：设定要链接到的文件名称，为必选项。若文件与页面不在同一个目录，需要加上适当的路径，一般路径格式为 href="域名或 IP 地址/文件路径/文件名#锚点名称"。

id 属性：用来定义页面内创建的锚点，在实现页面内部链接的时候使用。

target 属性：设定链接目标网页所要显示的视窗，默认为在当前窗口打开链接目标。可

选值为：_blank，_parent，_self，_top 及窗体名称。

- target="_blank"表示将链接的目标内容，在新的浏览器窗口中打开。
- target="_parent"表示将链接的目标内容，在父浏览器窗口中打开。
- target="_self"表示将链接的目标内容，在本浏览器窗口中打开(默认值)。
- target="_top"表示将链接的目标内容，在顶级浏览器窗口中打开。
- target="窗体名称"表示常用于框架或浮动框架中，将链接的目标内容，在"窗体名称"的框架窗体中打开，框架窗体名称已经事先在框架或浮动框架标记中命名。

例如：

```
<a href="http://www.taobao.com"  target="_selt" > 淘宝网</a>
```

这一段代码运行后，单击这个超链接，会在本窗口访问淘宝网。

2. 超链接的形式

XHTML 支持的超链接有以下几种形式：不同网页之间的跳转、链接至电子邮件、链接跳转到具体的锚点等。不同的超链接形式有不同的格式，具体如下。

- 链接到其他网页，基本格式：

```
<a  href="URL"> 锚点 </a>
```

在此处表示链接的是指定网页。运行时单击链接，转向另一个页面。

- 链接到图像上，基本格式：

```
<a  href="image_name.jpg" >锚点 </a>
```

运行时，单击超链接，跳转向一幅图片。

- 链接到电子邮件，基本格式：

```
<a  href="mailto:邮件地址"> 锚点 </a>
```

邮件地址形式为：name@site.come。

例如：

```
<a href="mailto: administrator@sohu.com" > 与搜狐网管理员联系</a>
```

运行后，单击超链接"与搜狐网管理员联系"，跳转到向管理员邮箱发信的页面。

- 页内链接：有的页面文本内容很多，浏览器打开页面往往从页面顶端开始显示，若用户需要的信息不在页面的起始部分，用户将费时费力地从上向下进行搜索。在此时，设置页内的链接是很有必要的。

实现页面内的链接时，需要先使用 id 属性定义一个锚点，格式为：

```
<a  id="锚点名称">预被链接后显示的首部分</a>
```

然后再使用 href 属性指向该锚点，格式为：

```
<a  href="#锚点名称"> </a>
```

#号表示链接目标与 a 标记属于同一个页面。

【例 2-4】建立 ASP.NET 页面，名称为 hyperlink.aspx。

其主体部分代码如下：

```
<body>
    百度网站的超级链接:
    <a href="http:\\www.baidu.com"> 百度搜索</a><br />
</body>
</html>
```

在浏览器中查看，运行结果如图 2-4 所示。

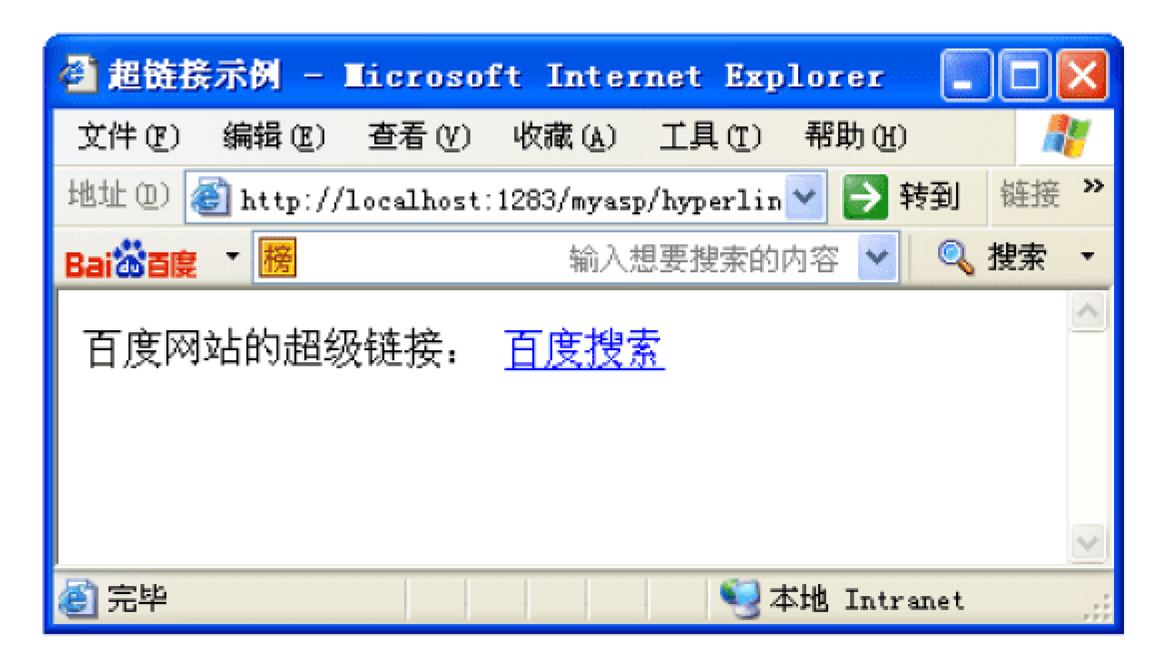


图 2-4　对页面使用超链接标记

【例 2-5】建立 ASP.NET 页面，名称为 hyperlink2.aspx。

其主体部分代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
    第 2 章  XHTML 基础知识
    <ul>
        <li>2.1 web 基本技术</li>
        <li><a href="#html">2.1.1 HTML</a></li>
        <li><a href="#xml">2.1.2 XML</a></li>
        <li><a href="#xhtml">2.1.3 XHTML</a></li>
     </ul>
   <!--在网页头部设定指向锚点的超链接-->
        <p>
         <a id="html">2.1.1   HTML</a><br/>
         <!--创建锚点 html-->
         HTML(HyperText Markup Language)超文本标记语言，是制作网页文档的.....<br/>
        </p>
        <p>
         <a id="xml">2.1.2   XML </a><br/>
          <!--创建锚点 xml-->
```

```
        HTML 是很成功的标记语言，......<br/>
      </p>
      <p>
        <a id="xhtml">2.1.3   XHTML </a><br/>
        <!--创建锚点 xhtml-->
        XML 虽然数据转换能力强大，.....<br/>
      </p>
    </div>
  </form>
</body>
```

以上代码运行结果如图 2-5 所示。



图 2-5　对页面内部使用超链接标记

2.3.7　图像标记<img />

Web 页面中的图像可以使网页更加生动、直观。常见的图像格式有 GIF、JPEG 和 PNG 等。其中，GIF 和 JPEG 格式能被大多数浏览器所支持。网页中的图像一般使用 72dpi 分辨率、RGB 色彩模式，在 XHTML 中使用<img />标记来向页面中插入图像。

图像标记语法格式为：

```
<img  src="图像文件名"  [alt="提示文本"]  [border="边框宽度"]  [align="对齐方式"]  [width="
宽度像素大小"]  [height="高度像素大小"]  alt="说明">
```

src：这个属性是必需的，用来链接图像的来源。若图像文件与 XHTML 页面文件处于同一目录下，则只写文件名称；若图像文件与页面不在同一目录，需要加上合适的路径，相对路径和绝对路径均可。

align：设置图像旁边文字的位置。可以控制文字出现在图片的上方、中间、底端、左侧和右侧。可选值为 top、middle、bottom、left 和 right，默认值为 bottom。

alt：区别于 HTML，每个图片标记必须有 ALT 说明文字。若用户使用文字浏览器，由于浏览器不支持图像，这些文字会替代图像显示出来；若用户使用支持图像显示的浏览器，

当鼠标移动至图像上时这些文字也会显现出来。

【例 2-6】在 d:\myasp\下建立新文件夹 images，将准备好的图像放入 images 文件夹，此处放入图片 fish.jpg。建立 ASP.NET 页面，名称为：picture.aspx。

XHTML 主体部分代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
        <img src="images/fish.jpg" align="left" width="200" height="150" alt="fish" /> 图片左对齐，
长 200 像素，宽 150 像素
    </div>
    <p>
    </p>
    <div align="center">
        <img / src="images/fish.jpg" align="middle"   width="100" height="100" alt="fish" /> 图片居
中，长 100 像素，宽 100 像素<br />
    </div>
    </form>
</body>
```

在浏览器中查看运行结果，如图 2-6 所示。



图 2-6　对页面使用图片标记

注意：当鼠标移至图片上时，会显示出 alt 属性的内容 fish。

实验 2　XHTML 基础知识

1. 实验目的

(1) 熟悉并会使用 XHTML 的页面格式及语法规则。

(2) 掌握 XHTML 文本标记、列表标记、表格标记、图像标记、超链接标记等标记的使用。

2. 实验内容和要求

(1) 创建一个 XHTML 页面 rumengling.aspx，输出结果如图 2-7 所示。

(2) 练习列表标记 songci.aspx，输出结果如图 2-8 所示。



图 2-7　rumengling.aspx 输出结果

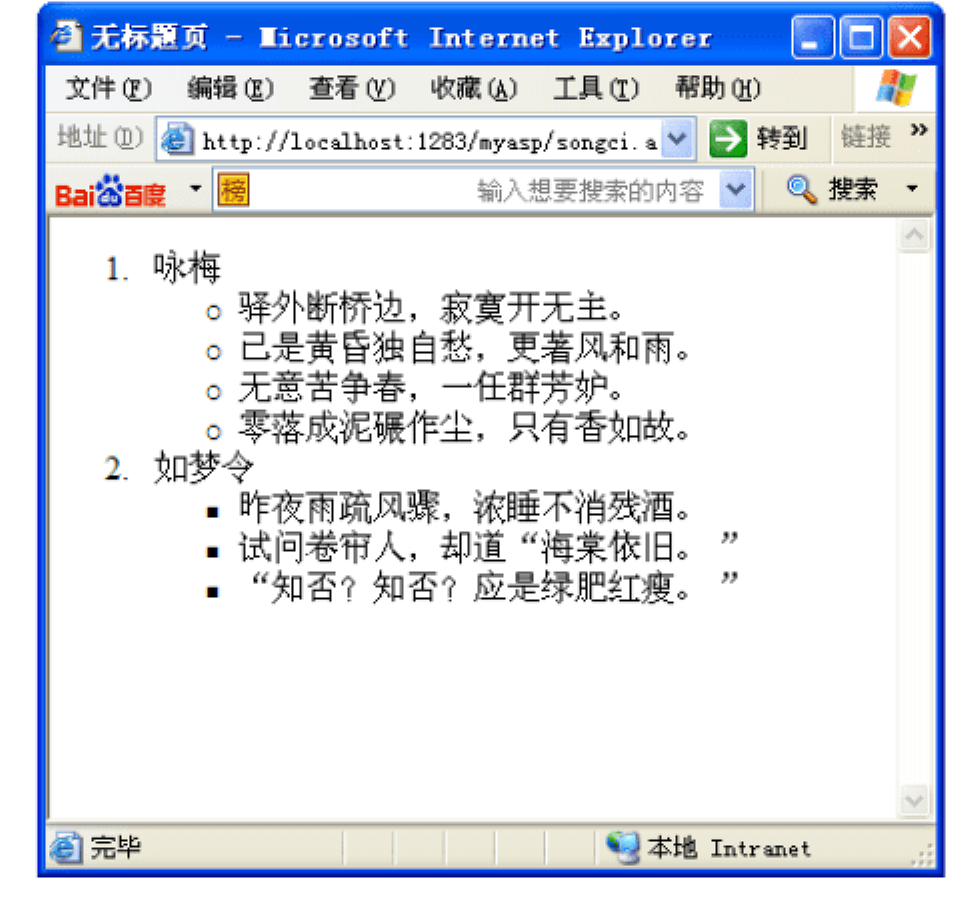


图 2-8　songci.aspx 输出结果

(3) 用 XHTML 语言编写符合以下要求的页面：标题为 An example of image，在浏览器窗口中显示一个图像。图像的宽度为 200 像素，高度为 150 像素，边框宽度为 10 像素。

(4) 在网页中做一张课程表，要求：所有的文字均居中；背景为黄色；表格居中，宽度为 500 像素；单元格间距与单元格边距为 20 像素。

(5) 创建一个网页，通过链接可以跳转到实验(1)创建的 rumengling.aspx 上。

第3章　ASP.NET基础知识

本章是学习 ASP.NET 非常关键的一章，是从认识了解到使用 ASP.NET 的一个关键点。在用 ASP.NET 进行程序开发时，其代码组织模式和页面的对象模型往往会贯穿整个开发过程，了解其基本运作方式有利于项目的顺利开发。

本章主要介绍 ASP.NET 的一些基础知识，学习和掌握这些知识是以后进行 ASP.NET 程序开发的基础和前提，本章正是围绕这些知识点进行介绍，首先介绍 ASP.NET 的页面框架和页面类，紧接着介绍了对 ASP.NET 各种程序文件进行存放的文件夹，接下来介绍 ASP.NET 的核心对象，主要包括 Response、Request、Application、Session、Server 等对象以及 ViewState 等，最后介绍了 ASP.NET 的配置文件 web.config 和全局文件 Global.asax。

本章的学习目标

- 了解 ASP.NET 页面的运行机制和页面的生命周期
- 了解 ASP.NET 所包含的文件夹以及各个文件夹的功能
- 掌握定义、创建和使用常用内置对象的方法
- 了解 Cookie 的基础知识以及设置和检索 Cookie 的方法
- 掌握 ASP.NET 的配置管理方式

3.1　页面管理

ASP.NET 页面是带.aspx 扩展名的文本文件，可以被部署在 IIS 虚拟目录树之下。页面由代码和标签(tag)组成，它们在服务器上动态地编译和执行，为提出请求的客户端浏览器(或设备)生成显示内容。对于 Web 开发人员来说，如果想提高页面的运行效率，首先需要了解 ASP.NET 页面是如何组织运行的。

3.1.1　ASP.NET 页面代码模式

ASP.NET 的页面包含两个部分：一部分是可视化元素，包括标签、服务器控件以及一些静态文本等；另一部分是页面的程序逻辑，包括事件处理句柄和其他程序代码。ASP.NET 提供两种模式来组织页面元素和代码：一种是单一文件模式；另一种是后台代码模式。两种模式功能是一样的，可以在两种模式中使用同样的控件和代码，但要注意使用的方式不同，接下来分别进行介绍。

1. 单一文件模式

在单一文件模式下，页面的标签和代码在同一个.aspx 文件中，程序代码包含在<script runat="server"></script>的服务器程序脚本代码块中间，并且代码中间可以实现对一些方法和属性以及其他代码的定义，只要在类文件中可以使用的都可以在此处进行定义。运行时，单一页面被视为继承 Page 类。

2. 后台代码模式

后台代码页面模式将可视化元素和程序代码分别放置在不同的文件中，如果使用 C#，则可视化页面元素为.aspx 文件，程序代码为.cs 文件，根据使用语言的不同，代码后缀也不同，这种模式也被称为代码分离模式。

ASP.NET 3.5 后台代码分离模式有很大的改进，简单易用且健壮性强，一个典型的代码分离模式的例子如下：

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public partial class _Default : System.Web.UI.Page
{
        protected void Page_Load(object sender, EventArgs e)
        {
            labContent.Text = Request.ServerVariables["ALL_HTTP"];
        }
}
```

ASP.NET 的代码分离模式，把一个程序文件分为一个.aspx 文件和一个对应的.aspx.cs 文件，前者是界面代码(主要用 html 编写)，后者是一些控制代码(主要用 c#编写)。.aspx 文件顶部的页面设置把两个文件联系在一块。在进行程序设计过程中，每一个控件都可以来触发事件。这些事件的代码单独在一个文件中，而网页的页面设计在单独的一个文件中，两个基本上是分离的，这样可以使代码文件更简洁。

3.1.2　页面往返机制

ASP.NET 网页是作为代码在服务器上运行的，因此，要使页面中的按钮或其他内容得到

处理，必须将该信息提交到服务器。每次页面提交时，都会在服务器端运行其代码，然后把运行的结果呈现给用户。ASP.NET 页面的处理循环如下。

(1) 用户通过客户端浏览器请求页面，页面第一次运行，执行初步处理。程序员可以通过编程对页面进行初始化等操作。

(2) 执行的结果以标记的形式呈现给浏览器，浏览器对标记进行解释并显示。

(3) 用户键入信息或从可选项中进行选择，或者单击按钮。如果用户单击的是一个超链接而不是按钮，页面就会定位到另一个页面，而该页面不会被进一步处理。

(4) 页面发送到 Web 服务器，在 ASP.NET 中称此为“回发”，也就是说页面发送回其自身。例如用户正在访问 Default.aspx 页面，则单击该页面上的某个按钮可以将该页面发送回服务器，发送的目的还是 Default.aspx。

(5) 在 Web 服务器上，该页再次运行，并且使用用户输入或选择的信息。

(6) 服务器将执行操作后的页面以 HTML 或 XHTML 标记的形式发送到客户端的浏览器。

只要用户访问同一个页面，该循环就会继续。用户每次单击按钮时，页面中的信息就会发送到 Web 服务器，然后该页面再次运行。每个循环称为一次“往返行程”。由于页面处理发生在 Web 服务器上，因此页面可以执行的每个操作都需要一次到服务器的往返行程。

有时可能需要代码仅在首次请求页时执行，而不是每次回发时都执行，这时候可以使用 Page 对象的 IsPostBack 属性来避免对往返行程执行不必要的处理。

3.1.3　页面生存周期

ASP.NET 页面在运行时将经历一个生命周期，在生命周期中，该页面将执行一系列处理步骤。这些步骤包括初始化、实例化控件、还原和维护状态、运行事件处理程序代码以及进行呈现。

与桌面应用程序中的窗体不同，ASP.NET 网页在使用窗体时不会启动或运行，并且仅当用户单击“关闭”按钮时才会卸载。这是由于 Web 具有断开连接的天性。浏览器从 Web 服务器请求页面时，浏览器和服务器相连的时间仅够处理请求。Web 服务器将页面呈现到浏览器之后，连接即终止。如果浏览器对同一 Web 服务器发出另一个请求，则即使是对同一个页面发出的，该请求仍会作为新请求来处理。

Web 这种断开连接的天性决定了 ASP.NET 页的运行方式。用户请求 ASP.NET 网页时，将创建该页的新实例。该页执行其处理，将标记呈现到浏览器，然后该页被丢弃。如果用户单击按钮以执行回发，将创建该页的一个新实例；该页执行其处理，然后再次被丢弃。这样，每个回发和往返行程都会导致生成该页的一个新实例。

3.2　特殊文件夹

开发者在对程序进行设计时应该将特定类型的文件存放在某些文件夹中，以方便在今后开发中的管理和操作。ASP.NET 保留了一些文件名称和文件夹名称，程序开发人员可以直接

使用，并且还可以在应用程序中增加任意多个文件和文件夹。如图 3-1 所示，而无须每次在给解决方案添加新文件时重新编译它们。ASP.NET 3.5 能自动、动态地预编译 ASP.NET 应用程序，并为应用程序定义好一个文件夹结构，利用这些定义好的文件夹，就可以自动编译代码，在整个应用程序中访问应用程序主题，并在需要时使用全局资源。下面介绍这些定义好的文件夹及它们的工作方式。

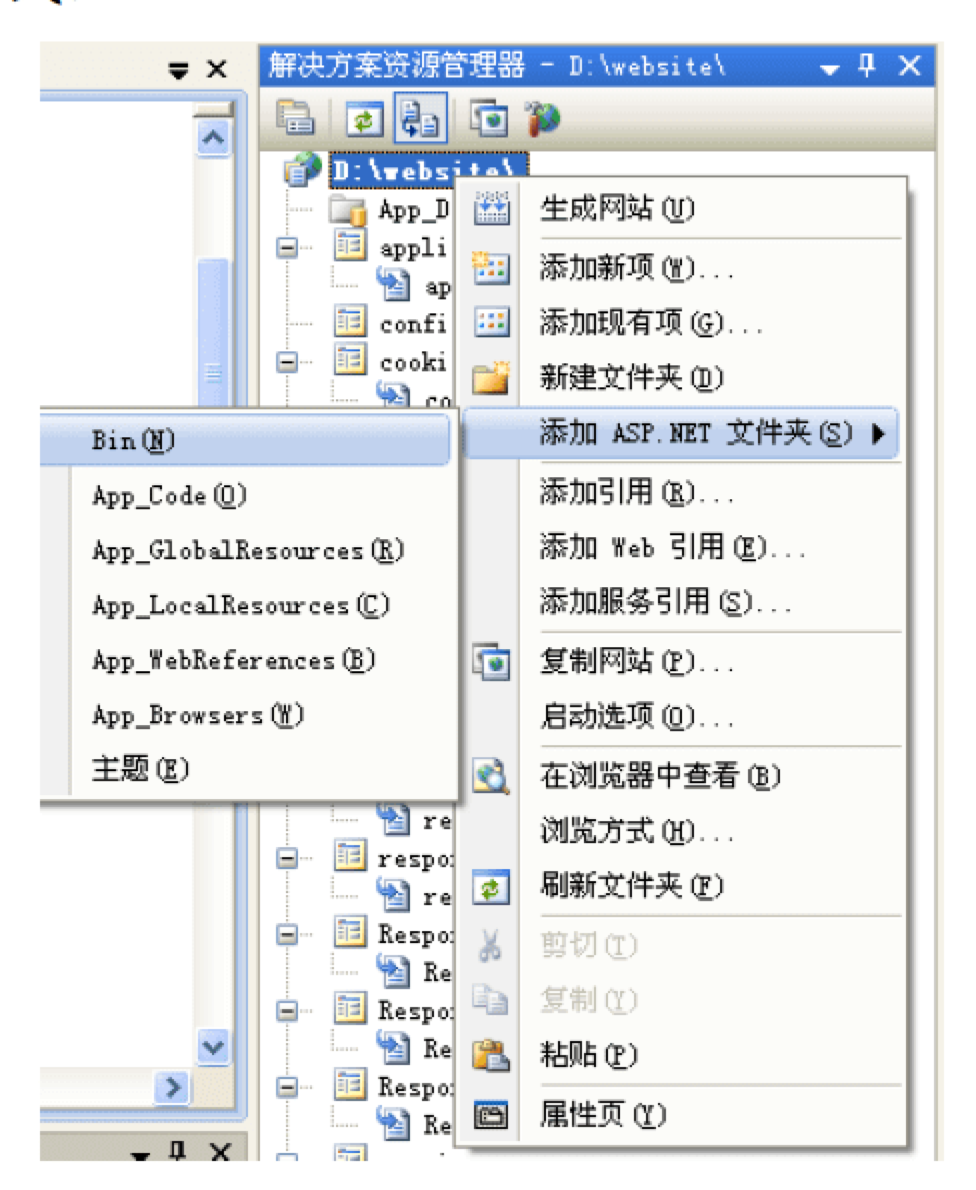


图 3-1　添加 ASP.NET 规定的特殊文件夹

3.2.1　App_Data 文件夹

App_Data 文件夹保存应用程序使用的数据库。它是一个集中存储应用程序所用数据库的地方。是 ASP.NET 为程序提供存储自身数据的默认位置，该文件夹内容不由 ASP.NET 处理。它通常以文件(诸如 Microsoft Access 或 Microsoft SQL Server 数据库、XML 文件、文本文件以及应用程序支持的任何其他文件)形式对数据进行存储。

注意：

默认 ASP.NET 账户被授予对文件夹的完全访问权限。如果碰巧要改变 ASP.NET 账户，一定要确保新账户被授予对该文件夹的读/写访问权。

3.2.2　App_Code 文件夹

App_Code 文件夹正好在 Web 应用程序根目录下，它存储所有应当作为应用程序的一部分动态编译的类文件。这些类文件自动链接到应用程序，而不需要在页面中添加任何显式指令或声明来创建依赖性。App_Code 文件夹中放置的类文件可以包含任何可识别的 ASP.NET 组件——自定义控件、辅助类、build 提供程序、业务类、自定义提供程序和 HTTP 处理程序等。

在开发时，对 App_Code 文件夹的更改会导致整个应用程序的重新编译。对于大型项目，这可能不受欢迎，而且很耗时。为此，鼓励大家将代码进行模块化处理到不同的类库中，按

逻辑上相关的类集合进行组织。应用程序专用的辅助类大多应当放置在 App_Code 文件夹中。

App_Code 文件夹中存放的所有类文件应当使用相同的语言。如果类文件使用两种或多种语言编写，则必须创建特定语言的子目录，以包含用各种语言编写的类。一旦根据语言组织这些类文件，就要在 web.config 文件中为每个子目录添加设置，关于 web.config 文件将在 3.4.1 节进行介绍。

3.2.3　Bin 文件夹

Bin 文件夹包含应用程序所需的，用于控件、组件或者需要引用的任何其他代码的可部署程序集。该目录中存在的任何.dll 文件将自动地链接到应用程序。可以在 Bin 文件夹中存储编译的程序集，并且 Web 应用程序任意处的其他代码会自动引用该文件夹。典型的示例是如果为自定义类编译好了代码，那么就可以将编译后的程序集复制到 Web 应用程序的 Bin 文件夹中，这样所有页都可以使用这个类。

Bin 文件夹中的程序集无需注册。只要.dll 文件存在于 Bin 文件夹中，ASP.NET 就可以识别它。如果更改了.dll 文件，并将它的新版本写入到了 Bin 文件夹中，则 ASP.NET 会检测到更新，并对随后的新页请求使用新版本的.dll 文件。下面介绍一下 Bin 文件夹的安全性。

将编译后的程序集放入 Bin 文件夹中会带来安全风险。如果是自己编写和编译的代码，那么设计者是了解代码的功能的。但是，设计者必须像对待任何可执行代码一样来对待 Bin 文件夹中已编译的代码。在完成代码测试并确信已了解代码功能之前，要对已编译的代码保持谨慎的态度。Bin 文件夹中程序集的作用范围为当前应用程序。因此，它们无法访问当前 Web 应用程序之外的资源或调用当前 Web 应用程序之外的代码。此外应该注意，在运行时，程序集的访问级别由本地计算机上指定的信任级别确定。

App_Code 文件夹和 Bin 文件夹是 ASP.NET 网站中的共享代码文件夹，如果 Web 应用程序要在多个页之间共享代码，就可以将代码保存在 Web 应用程序根目录下的这两个特殊文件夹中的某个文件夹中。当创建这些文件夹并在其中存储特定类型的文件时，ASP.NET 将使用特殊方式进行处理。

3.2.4　其他文件夹介绍

1. App_Themes 文件夹

主题是为站点上的每个页面提供统一外观和操作方式的一种新方法。通过 skin 文件、CSS 文件和站点上服务器控件使用的图像来实现主题功能。所有这些元素都可以构建一个主题，并存储在解决方案的 App_Themes 文件夹中。把这些元素存储在 App_Themes 文件夹中，就可以确保解决方案中的所有页面都利用该主题，并把其元素应用于控件和页面的标记。

2. App_GlobalResources 文件夹

资源文件是一些字符串表，当应用程序需要根据某些事情进行修改时，资源文件可用于这些应用程序的数据字典。可以在这个文件夹中添加程序集资源文件(.resx)，它们会动态编译，成为解决方案的一部分，提供给程序中的所有.aspx 页面使用。在使用 ASP.NET 1.0/1.1

时，必须使用 resgen.exe 工具，把资源文件编译为.dll 或.exe，才能在解决方案中使用。而在 ASP.NET 3.5 中，资源文件的处理就容易多了。除了字符串之外，还可以在资源文件中添加图像和其他文件。

3. App_LocalResources 文件夹

App_GlobalResources 文件夹用于合并可以在应用程序范围内使用的资源。如果对构造应用程序范围内的资源不感兴趣，而对只能用于一个.aspx 页面的资源感兴趣，就可以使用\App_LocalResources 文件夹。可以把专用于页面的资源文件添加到\App_LocalResources 文件夹中，方法是构建.resx 文件名，如下所示：

```
Default.aspx.resx
Default.aspx.fi.resx
Default.aspx.ja.resx
Default.aspx.en-gb.resx
```

现在，可以从\App_LocalResources 文件夹的相应文件中检索在 Default.aspx 页面上使用的资源声明。如果没有找到匹配的资源，就默认使用 Default.aspx.resx 资源文件。

3.3　ASP.NET 的常用内置对象

在 ASP.NET 里面，有几大常用内置对象，即 Response、Request、Application、Session、Server、ViewState、Cookie 等。下面将分别介绍一下这些对象的常用属性及方法。

3.3.1　Response 对象

Request 对象与 Response 对象就像一般程序语言里的 Input 及 Output 命令(或函数)，若要让 ASP.NET 程序能够接收来自前端用户的信息，或者想将信息传递给前端，都必须依赖这两个对象。简言之，Request 对象掌管 ASP.NET 的 Input 功能，而 Response 对象则负责 Output 功能。

Response 对象实际是在执行 system.web 命名空间中的类 HttpResponse。CLR 会根据用户的请求信息建立一个 Response 对象，Response 将用于回应客户端浏览器，告诉浏览器回应内存的报头、服务器端的状态信息以及输出指定的内容。Response 的常用属性和方法如表 3-1 和表 3-2 所示。

表 3-1　Response 对象常用的属性列表

属　　性	说　　明
Buffer	获取或设置是否缓冲输出
Cache	获取缓存信息
CharSet	获取和设置输出流的 HTTP 字符集
ContentType	获取和设置输出流的 MIME 类型

(续表)

属　　性	说　　明
Cookie	获取 Cookie 集合
Expires	获取和设置浏览器缓存超时时间
IsClientConnected	获取客户端是否和服务器连接
Status	设置返回给客户端的状态
StatusCode	获取和设置返回给客户端状态字符串
StatusDescription	获取和设置状态说明

表 3-2　Response 对象常用方法列表

方　　法	说　　明
AddHeader	添加 HTTP 头信息
AppendCookie	添加一个 Cookie
AppendHeader	添加 HTTP 头信息
AppendToLog	添加自定义信息到 IIS 日志中
BinaryWrite	以二进制的方式输出
Clear	清除输出缓存
Close	关闭和客户端的 Socket 连接
End	发送所有缓冲到客户端，并且停止执行页面
Flush	发送所有缓存到客户端
Redirect	重新定向 URL
SetCookie	更新一个已有的 Cookie
Write	输出信息
WriterFile	直接将指定文件写到输出流

下面举例说明该对象最常用的一些属性和方法。

- Response.ContentType()方法

ContentType 属性设定 Response 对象的输出内容类型，默认值为：text/html。例如：

```
Response.ContentType= "image/gif "
```

表示输出内容类型为 GIF 图形文件。

- Response.Expires=number

Expires 属性指定了在浏览器上缓冲存储的页距过期还有多少时间。如果用户在某个页过期之前又回到此页，就会显示缓冲区中的版本。若此属性在一页上设置了多次，则使用最短的时间。如指定了日期(Date)而未指定时间(Time)，页面在午夜失效。如指定了时间而未指定日期，则在脚本执行的当天那个时间失效。例如：

```
Response.ExpiresAbsolute=DateTime.Now
```

表示页面在当前时间失效，即马上过期。

● Response.Buffer=Flag

Buffer 属性表示是否对页面输出进行缓冲。如有缓冲，服务器在所有当前处理的页面的语句被处理之前不将 Response 送往客户端，除非有 Flush 或 End 方法被调用。参数 Flag 表示是否对页面输出进行缓冲。True 表示需要，False 表示不需要，默认值是 True。

● Response.Flush()方法

立即将缓冲区中页面输出。

● Response.Write()方法

Write 方法输出指定的文本内容。例如：Response.Write("Hello World! ")。

● Response.WriteFile()方法

直接将输出的内容写入一个文件中，例如：Response.WriteFile("c:\\write.txt")。

【例 3-1】演示 Response.Write()方法的使用。

(1) 新建一个名为 webside 的网站，在其中添加一个名为 Response.aspx 的网页。

(2) 在 Response.aspx.cs 页面中添加下面代码。

```
protected void Page_Load(object sender, EventArgs e)
  {
      Response.Write("Hello World!");
      Response.Write("<h2>Hello World!</h2>");
      Response.Write("<p style='color:#0000ff'>Hello World!</p>");
  }
```

(3) 程序运行效果如图 3-2 所示。



图 3-2　Response.aspx 运行效果

3.3.2　Request 对象

Request 对象是 ASP.NET 当中最有用的对象之一，它与 Response 对象一起使用，达到沟通客户端及服务器端的作用，使它们之间可以很简单地交换数据，由此可见该对象的重要。Request 对象接收客户端通过表单或者 URL 地址串发送来的变量，同时，也可以接收其他客户端的环境变量，比如浏览器的基本情况、客户端的 IP 地址等。所有从前端浏览器通过 HTTP 通信协议送往后端 Web 服务器的数据，都是借助 Request 对象完成的，总而言之，Request 对象含有用户端最为丰富的数据。

Request 对象常用的属性和方法如表 3-3 和表 3-4 所示。

表 3-3　Request 对象常用属性列表

属　　性	说　　明
ApplicationPath	获得 ASP.NET 应用程序虚拟目录的根目录
Browser	获取和设置客户端浏览器的兼容性信息
ContentLength	客户端发送信息的字节数
ContentType	获取和设置请求的 MIME 类型
Cookies	获取客户端 Cookie
FilePath	当前请求的虚拟路径
Files	获取客户端上传的文件集合
Form	获取表单变量集合
Headers	获取 HTTP 头信息
HttpMethod	HTTP 数据传输方法，例如 GET、POST
Path	获取当前请求的虚拟路径
PhysicalPath	获取请求的 URL 物理路径
QueryString	获取查询字符串集合
ServerVariables	获取服务器变量集合
TotalBytes	获取输入文件流的总大小
Url	获取当前请求的 URL
UrlReferrer	获取该请求的上一个页面
UserAgent	客户端浏览器信息
UserHostAddress	客户端 IP 地址
UserHostName	客户端 DNS 名称
UserLanguages	客户端语言

表 3-4　Request 对象方法列表

名　称	说　明
BinaryRead	以二进制方式读取指定字节的输入流
MapPath	影射虚拟路径到物理路径
SaveAs	保存 HTTP 请求到硬盘
ValidateInput	验证客户端的输入是否存在危险的数据

ASP.NET 是使用表单(Form)来实现用户数据提交的。对于 HTML 表单，可以使用 Get 方法或 Post 方法来实现数据提交。如果使用 Get 方法，就要使用 Request 对象的 QueryString 集合来得到相关的信息；如果使用 Post 方法，就要使用 Request 对象的 Form 集合来得到相关信息。下面分别讲解如何使用 Get 方法和 Post 方法。

● Get 方法

使用 Get 方法进行数据提交需要使用 Request.QueryString 方法来取得传递的数据，传输的数据量比较小，一般限制在 2KB 左右，用户要提交的信息往往是作为查询字符串加在 URL 的后面传给接收程序。URL 是指 Uniform Resource Locator (统一资源定位器)，它的通常形式是类似 http://www.abc.com 形式的字符串。而查询字符串则是加在 URL 的后面，与前面的 URL 一起提交给服务器的字符串。例如：http://www.domain.com/test.aspx?name=myname&password=mypassword。

● Post 方法

Post 方法是用 Request 对象的 Form 集合属性来得到用户提交的数据。使用该方法时，用户浏览器的地址栏中不会显示相关的查询字符串。因此，如果需要提交的数据很多时，应使用 Post 方法，因为它对数据的大小和长度没有什么限制。另外，如果使用 Web 表单而不是 HTML 表单的话，那么也只能使用 Post 方法来获取数据，因为这是 Web 表单默认使用的方法，无需在定义表单时说明。

与 Get 方法相比，Post 方法对数据的大小和长度没有什么限制，所以对于大量的数据提交，使用 Post 方法是十分有利的。另外，由于地址栏中不显示相关的查询字符串，那么使用 Post 方法就十分适合用来传递保密信息，例如用户的账号和密码等。

【例 3-2】演示利用 Request 对象的 Form 集合来得到相关信息的方法。

(1) 在 webside 网站中添加一个名为 Request1.aspx 和名为 Request2.aspx 的网页，在 Request1.aspx 页面的<body>标签中添加如下代码：

```
<form id="form1" runat="server">
<div>
    <a href="Request2.aspx?username=chen">Example</a>
</div>
</form>
```

(2) 在 Request2.aspx.cs 页面中添加下面代码。

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    Response.Write(Request.QueryString);
      Response.Write("<br/>");
    Response.Write(Request.QueryString["username"]);
}
```

(3) 程序运行效果如图 3-3 和图 3-4 所示。

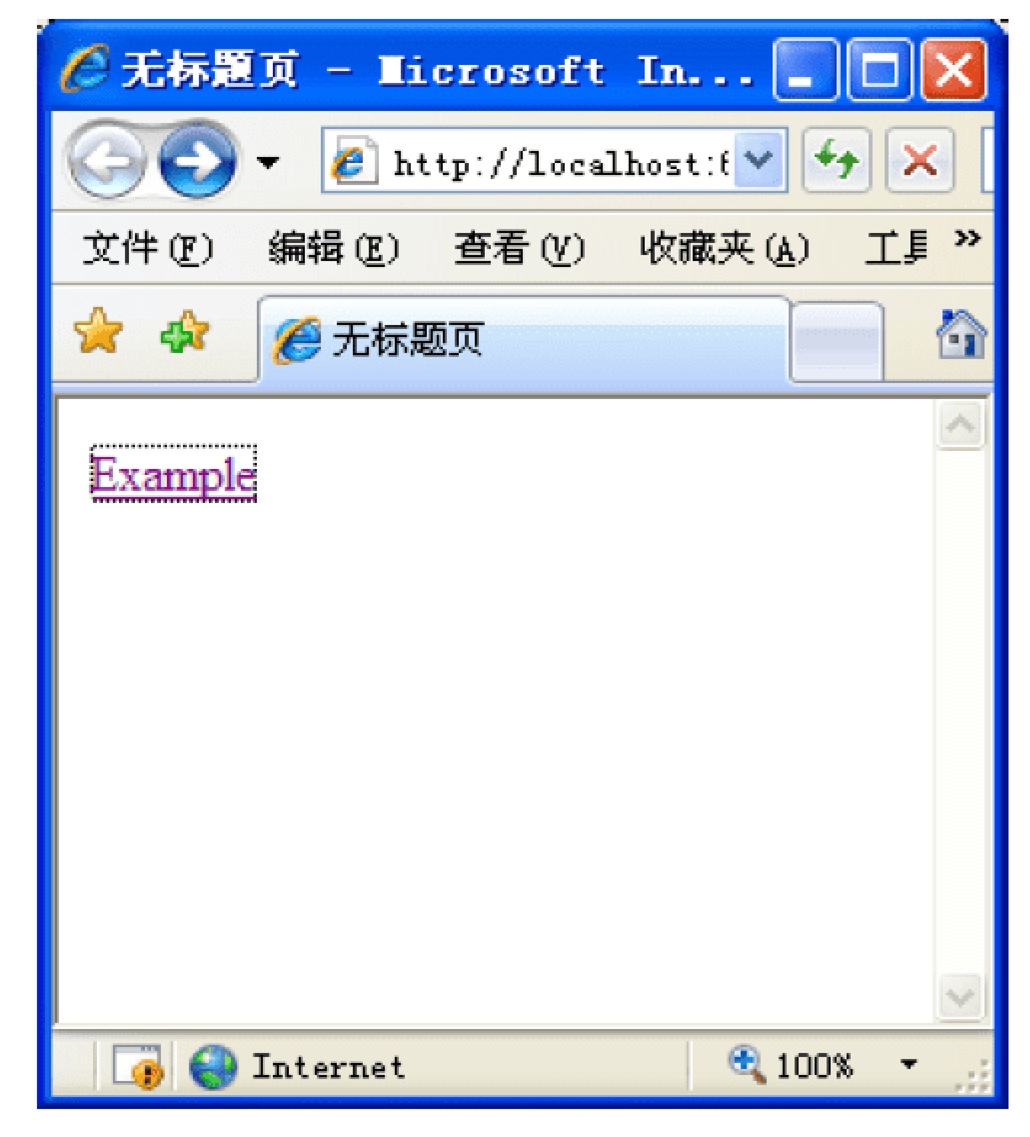


图 3-3　Request1.aspx 运行效果

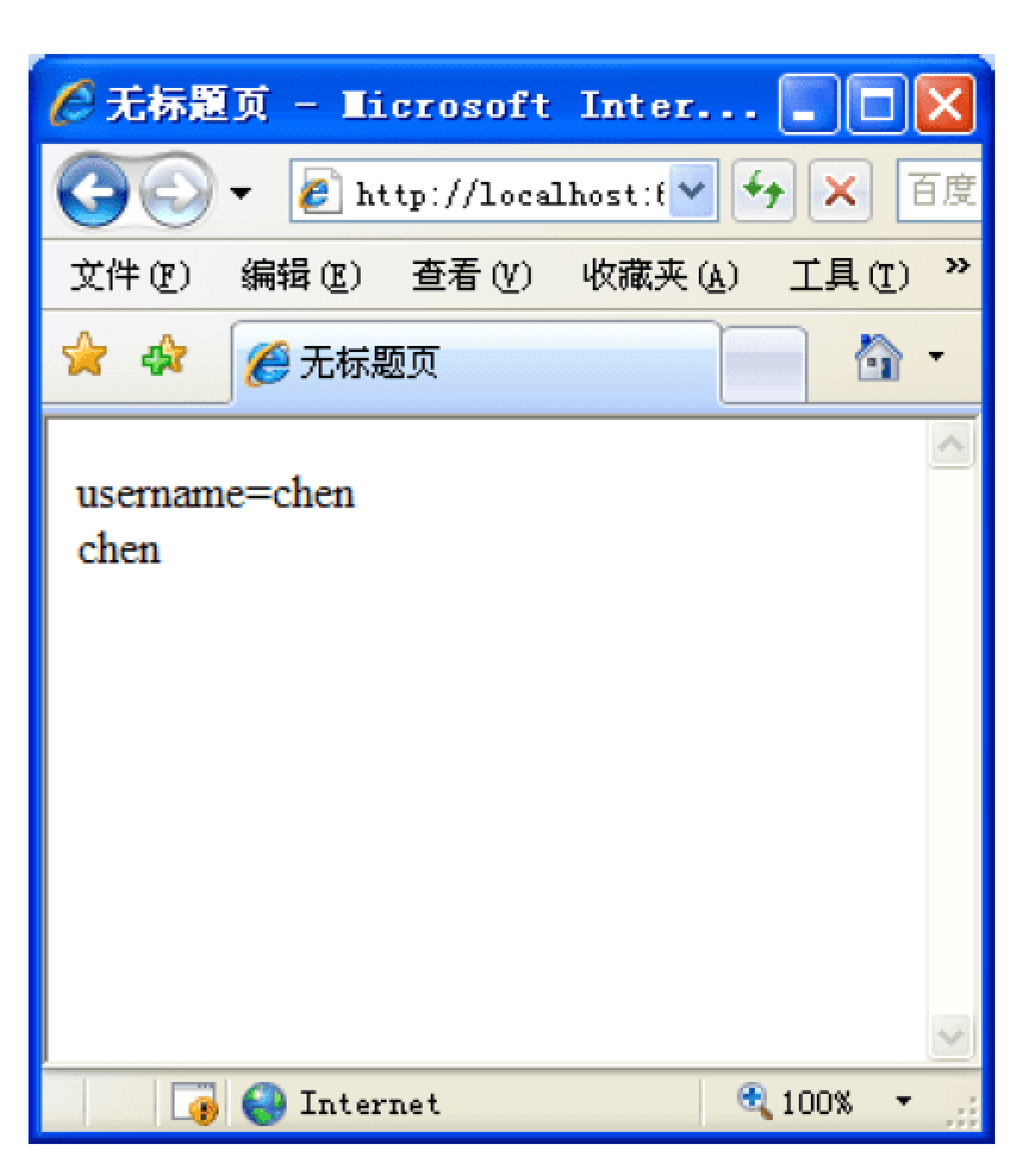


图 3-4　Request2.aspx 运行效果

3.3.3　Application 对象

人们最常用的保存变量的内容的方法是使用文件，但是毕竟对文件的操作是比较麻烦的事情，有没有更简单的方法呢？其中一种比较简单的方法就是使用 Application 对象来保存希望传递的变量。由于在整个应用程序生存周期中，Application 对象都是有效的，所以在不同的页面中都可以对它进行存取，就像使用全局变量一样方便。

在 ASP.NET 环境中，Application 对象来自 HttpApplicationState 类，它可以在多个请求、连接之间共享公用信息，也可以在各个请求连接之间充当信息传递的管道，Application 对象常用的属性和方法如表 3-5 和表 3-6 所示。

表 3-5　Application 属性列表

属　性	方　法
AllKeys	获得访问 HttpApplicationState 集合的所有键
Contents	获得 HttpApplicationState 对象的引用
Count	获得 HttpApplicationState 集合的数量
Item	通过名称和索引访问 HttpApplicationState 集合
Keys	获得访问 HttpApplicationState 集合的所有键，从 NameObjectCollectionBase 继承
StaticObjects	获得所有使用<object>标签声明的应用程序集对象

表 3-6　Application 方法列表

方　法	说　明
Add	添加一个新的对象到 HttpApplicationState 集合
Clear	清除 HttpApplicationState 集合中的所有对象
Get	通过索引和名字获得 HttpApplicationState 对象
GetKey	通过索引获得一个 HttpApplicationState 名称
Lock	锁定访问 HttpApplicationState 变量
UnLock	取消锁定，一般情况下需要操作 Application 变量则设置为 Lock，操作完成后则设置为 Unlock
Remove	从 HttpApplicationState 集合删除一个对象
RemoveAll	删除 HttpApplicationState 集合所有对象
RemoveAt	根据索引删除一个 HttpApplicationState 对象
Set	更新一个 HttpApplicationState 变量

下面对该对象常用的一些属性和方法进行说明。

- Add 方法，加入一个对象到 Application 对象的 Stat 集合中。例如：

```
Application.Add("string1","test")
```

表示向 Application 的 Stat 集合中加入一个名为 string1 的值为 test 的字符串，其实它的效果和 Application("string1")="test"以及 Application.Item("sting1")="test"是一样的。

- Remove 方法，根据给出的标识从 Application 对象的 Stat 集合中删去，例如：

```
Application.Remove("sting1")
```

表示把标识为 sting1 的共享对象 sting1 从 Application 对象的 Stat 集合中删去。使用它可以清除用 Add 方法添加的对象。

- RemoveAll 方法，把 Application 对象 Stat 集合中的所有对象清除，在对属性的使用举例中，已经见过了它的用法，但是值得小心，不提倡使用它，因为在编程中并不清楚是否有其他页面要依赖于某个 Application 的公用变量，一旦清除将造成不可预知的错误。
- Clear 方法，作用和 RemoveAll 方法一样。
- Get 方法，允许使用名字标识或者是下标来取得 Application 对象 Stat 集合中的对象元素。例如：

```
Tmp=Application.Get(0)
```

- Set 方法，修改 Application 对象 Stat 集合中指定标识所对应的对象值。例如：

```
Application("sting1","try")
```

表示把开始为 sting1 变量设置的值 test 改为 try。

【例 3-3】在线统计网站访问人数，演示 Application 对象一些方法的使用。

(1) 在 webside 网站中添加一个名为 Application.aspx 的网页。

(2) 在 Application.aspx.cs 页面中添加下面代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    Application.Lock();
    Application["usercount"] = (Convert.ToInt32(Application["usercount"]) + 1).ToString();
    Application.UnLock();
    labContent.Text = Application["usercount"].ToString();
}
```

(3) 程序运行效果如图 3-5 所示。

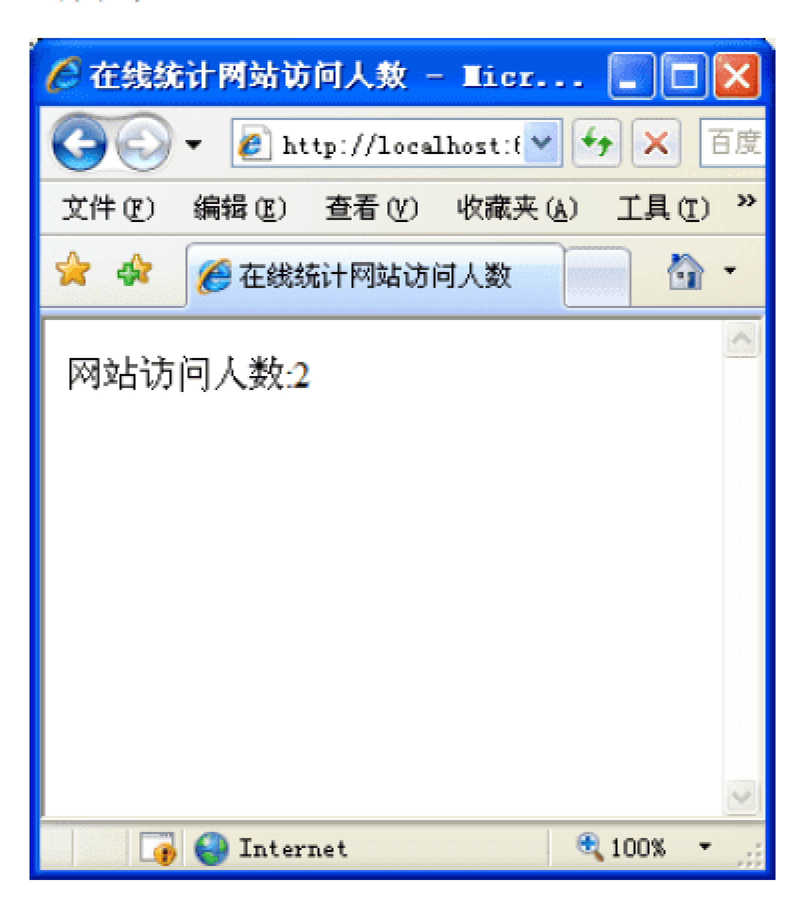


图 3-5　Application.aspx 运行效果

3.3.4　Session 对象

Session 对象的作用也是用于储存特定的信息，但是它和 Application 对象在储存信息所使用的对象是完全不同的。Application 对象储存的是共享信息，而 Session 储存的信息是局部的，是随用户不同而不同的。如果只需要在不同页中共享数据，而不是需要在不同的客户端之间共享数据就可以使用 Session 对象。

Session 的生命周期是有限的(默认值为 20 分钟)，它可以使用 Timeout 属性进行设置。在 Session 的生命周期内，Session 的值是有效的。如果用户在大于生命周期的时间里没有再访问应用程序，Session 就会自动过期，Session 对象将会被 CLR 释放，其储存的数据信息将永远不再存在。Session 对象的常用属性和方法如表 3-7 和表 3-8 所示。

表 3-7　Session 对象常用属性列表

方　　法	说　　明
CodePage	获得或设置字符集标识
Contents	获得当前 Session 状态对象的引用
CookieMode	获得当前的 Cookie 模式，以确定系统是否要将 Session 配置为不需要 Cookie 支持
Count	Session 状态集合的总数
IsCookieless	是否需要 Cookie 支持，如果需要就可以将 Session ID 保存在 Cookie 中，如果不需要就必须嵌入在 URL 中
IsNewSession	标志当前 Session 是否是新的 Session
IsReadOnly	是否只读
IsSynchronized	是否同步
Item	通过索引获得或者设置单个 Session 值
Keys	获得 Session 集合的所有键
LCID	获得和设置当前 Session 的本地标识符
Mode	获得当前的 Session 模式
SessionID	获得 Session 的唯一编号，为了区别不同的会话，系统会为每一个会话分配一个唯一的 ID
StaticObjects	获得在 Global.asax 中以<object Runat="Server" Scope="Session" />声明的对象集合
Timeout	获得和设置会话超时时间，如果客户端在连续一个时间段内没有反应，就自动清除会话，断开连接，Timeout 就是这个时间段

表 3-8　Session 对象常用方法列表

方　　法	说　　明
Add	添加一个新对象到 HttpApplicationState 集合
Clear	清除 HttpApplicationState 集合中的所有对象
Get	通过索引和名字获得 HttpApplicationState 对象
Abandon	清除当前会话
Copyto	复制 Session 状态集合到一个一维数组
Remove	从 HttpApplicationState 集合删除一个对象
RemoveAll	删除 HttpApplicationState 集合所有对象
RemoveAt	根据索引删除一个 HttpApplicationState 对象

【例 3-4】演示 Session 方法的使用。

(1) 在 website 网站中添加名为 Session1.aspx 和 Session2.aspx 的两个网页。

(2) 在 Session1.aspx.cs 页面中添加如下代码。

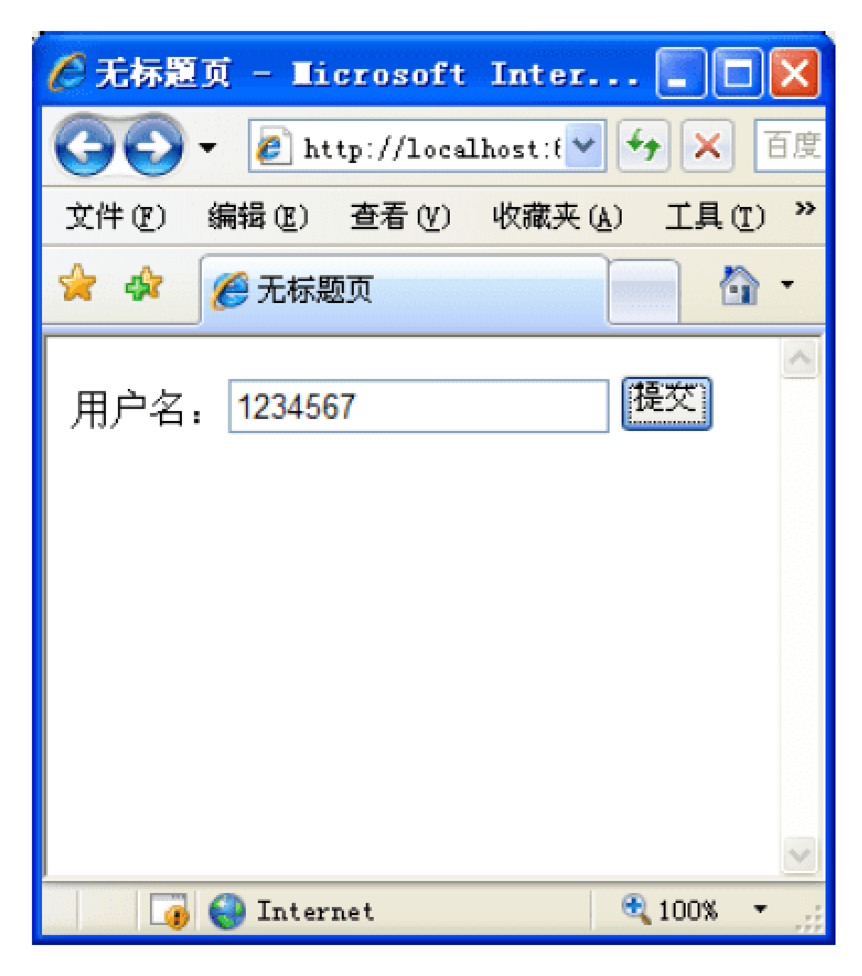
```
protected void Button1_Click(object sender, EventArgs e)
```

```
    {
        Session["username"] = TextBox1.Text;
        Response.Redirect("Session2.aspx");
    }
```

(3) 在 Session2.aspx.cs 页面的中添加如下代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["username"] != null)
        {
            Label1.Text = Session["username"].ToString();
        }
        else
        {
            //请输入用户名
            Response.Redirect("Session1.aspx");
        }
    }
```

(4) 程序运行效果如图 3-6 和图 3-7 所示，从图 3-6 和图 3-7 中可以看到利用 Session 对象可以存储用户的相关信息，并在需要时进行读取显示。



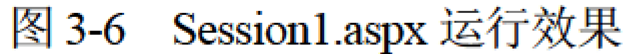


图 3-6　Session1.aspx 运行效果

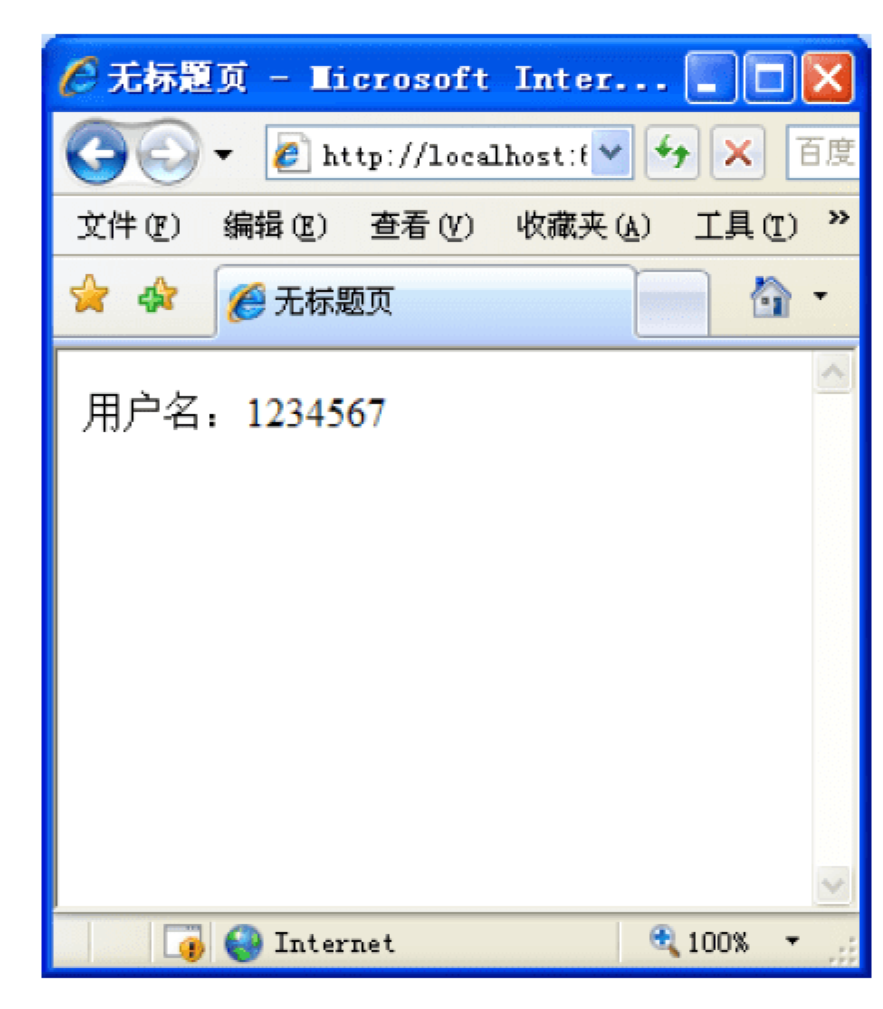


图 3-7　Session2.aspx 运行效果

3.3.5　Server 对象

Server 对象即服务器对象，就是在服务器上工作的一个对象，用于建立 COM 对象实例、处理应用程序错误，在页面之间传递控件，获取最新出错信息，对 HTML 进行编码和解码等。Server 对象提供许多访问的方法和属性帮助程序有序地执行，主要属性和方法如表 3-9 和表 3-10 所示。

表 3-9　Server 对象常用属性

名　称	说　明
MachineName	获得服务器计算机名称
ScriptTimeout	获得和设置请求超时的事件

表 3-10　Server 对象常用方法

方　法	说　明
CreateObject	建立一个 COM 组件对象的实例
Execute	执行指定资源并返回
Transfer	结束当前页执行，转到其他页执行
HtmlEncode	进行 HTML 编码
HtmlDecode	进行 HTML 解码
MapPath	对虚拟目录进行物理映射
UrlEncode	进行 URL 编码
UrlDecode	进行 URL 解码

Server 对象的一个重要功能是对字符进行 URL 和 HTML 的编码和解码。URL 编码的目的是保证所有浏览器能够正确地传输 URL 路径，一些特殊字符如？、&、/、空格和中文字符等，在传输时都有可能让浏览器发生错误。先通过编码再将其传输，在需要使用时又通过解码将其还原。HTML 编码的作用是将所有字符全部转化为 HTML 中能够用来显示的字符，例如<p>这样的字符如果直接显示就是一个段落，如果转换以后就会变成<p>浏览时就可以正确显示出<p>，就不会造成一些错误。下面对一些常用的方法进行解释。

● HtmlEncode、HtmlDecode 方法

HtmlEncode 方法的作用是将代码中的 HTML 标签进行了转码，目的是显示这段源代码，而不是去执行它。

例如当用户需要显示一个<HTML>时，如果写 Response.Write("<html>");的话，是不能达到目的的，需要这样来写才能达到目的：

```
Response.Write Server.HtmlEncode("<HTML>");
```

这时，查看源文件，就可以看到，使用该方法已经把<HTML>字符串转换成为<html>。但是，在显示时还是显示<HTML>，这样，使用 HtmlEncode 方法就可以让用户随心所欲地显示用户所要显示的内容，而不会和用户的页面混合在一起。

HtmlDecode 方法的作用与 HtmlEncode 方法的作用正好相反，它会将被 HTML 编码的代码解码，恢复代码本来面目。

● UrlEncode、UrlDeconde 方法

UrlEncode 方法与 HtmlEncode 方法类似，将指定的代码以 URL 格式进行编码，把 URL 中的所有特殊字符转化成为非功能的等价内容。通常在把 URL 作为查询字符串时使用。用

下面的方法可以看出这些特定的字符被转化成的内容：

```
Response.Write(Server.UrlEncode(http://www.abc.com));
```

执行完成后，显示的是：

```
http%3A%2F%2Fwww%2Eabc%2Ecom
```

UrlDecode 和 UrlEncode 刚好相反，该方法将 URL 格式代码进行解码。

● MapPath 方法

Server.MapPath 方法将虚拟路径转换为绝对路径。这种方法在需要包含或执行其他的文件并需要制定路径名，但路径名又常常发生变化的情况下使用。例如：

```
StingsPath=Server.MapPath("/");
```

● Transfer(path)方法

终止当前程序的执行，进入 path 所指的程序。该方法可以把控制传递出去，可以把原来页面的所有内置对象和这些对象的状态都传递给新的页面，比如 Request 对象的查询字符串。使用这种方法还可以把一个大的程序划分成小的模块，然后用 Transfer 方法把各个模块联系起来。

● Execute(path)方法

在当前程序中执行 path 所指定的程序，它与 Transfer 方法的不同之处在于，当 path 所指的程序运行完毕后，将继续执行当前程序中后面的代码。

3.3.6　ViewState(视图状态)对象

ViewState 对象是状态管理中常用的一种对象，可以用来保存页和控件的值，下面对其进行介绍。

1. ViewState 简介

视图状态是 ASP.NET 页框架默认情况下用于保存往返过程之间的页面信息以及控件值的方法。当呈现页的 HTML 形式时，需要在回发过程中保留的页的当前状态和值将被序列化为 Base64 编码的字符串，并输出到视图状态的隐藏字段中。通过实现自定义的 PageStatePersiste 类以存储页数据，也可以更改默认行为并将视图状态存储到另一个位置，如 SQL Server 数据库。

程序员可以通过使用页面的 ViewState 属性将往返过程中的数据保存到 Web 服务器端，然后利用自己的代码访问视图状态。ViewState 属性是一个包含密钥/值对(其中包含视图状态数据)的字典。

视图状态中存储的常见数据类型有：字符串、整数、布尔值、Array 对象、ArrayList 对象、哈希表和泛型对象等。

2. 使用 ViewState 时的注意事项

使用 ViewState 可以带来很多方便，但是也有一些问题是需要注意的。

(1) 视图状态提供了特定 ASP.NET 页面的状态信息。如果需要在多个页上使用信息，或者需要在访问网站时保留信息，则应当使用另一个方法(如应用程序状态、会话状态或个性化设置)来维护状态。

(2) 视图状态信息将序列化为 XML，然后使用 Base64 编码进行编码，这将生成大量的数据。将页回发到服务器时，视图状态信息将作为页回发信息的一部分发送。如果视图状态包含大量信息，则会影响页的性能。因此，建议使用一些典型数据来测试页性能，确定视图状态的大小是否是导致应用程序性能问题的“瓶颈”。

(3) 虽然使用视图状态可以保存页和控件的值，但是在某些情况下，需要关闭视图状态。比如使用 GridView 控件显示数据，单击 GridView 控件的下一页按钮，此时，GridView 控件呈现的数据已经不再是前一页的数据，那么如果使用视图状态将前一页数据保存下来，不仅没有必要而且还会生成大量隐藏字段，增大页面的体积，此时应当关闭视图状态以移除由 GridView 控件生成的大量隐藏字段。

另一个重要的考虑因素是，如果隐藏字段中的数据量过大，某些代理的防火墙将禁止访问包含这些数据的页。由于所允许的最大数据量随所采用的防火墙和代理的不同而不同，因此大量隐藏字段可能会导致偶发性问题。为了帮助避免这一问题，如果 ViewState 属性中存储的数据量超过了页的 MaxPageStateFieldLength 属性中指定的值，该页会将视图状态拆分为多个隐藏字段，可以使每个单独字段的大小在防火墙拒绝的大小以下。

【例 3-5】演示如何使用 ViewState 对象，从视图状态中读取值和将输入的新值保存在视图状态中，并可以进行读取显示。

(1) 在 webside 网站中添加一个名为 ViewState.aspx 的网页。

(2) 在 ViewState.aspx.cs 页面中添加下面代码。

```
public partial class Default2 : System.Web.UI.Page
{
    string VSString = "basketball";
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            ViewState.Add("favorite", VSString);
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        VSString = this.TextBox1.Text;
        ViewState["favorite"] = VSString;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        if (ViewState["favorite"] != null)
        {
```

```
                this.Label1.Text = ViewState["favorite"].ToString();
            }
            else
            {
                this.Label1.Text = "查看的 ViewState 值不存在";
            }
        }
    }
```

(3) 程序运行效果如图 3-8 和图 3-9 所示。

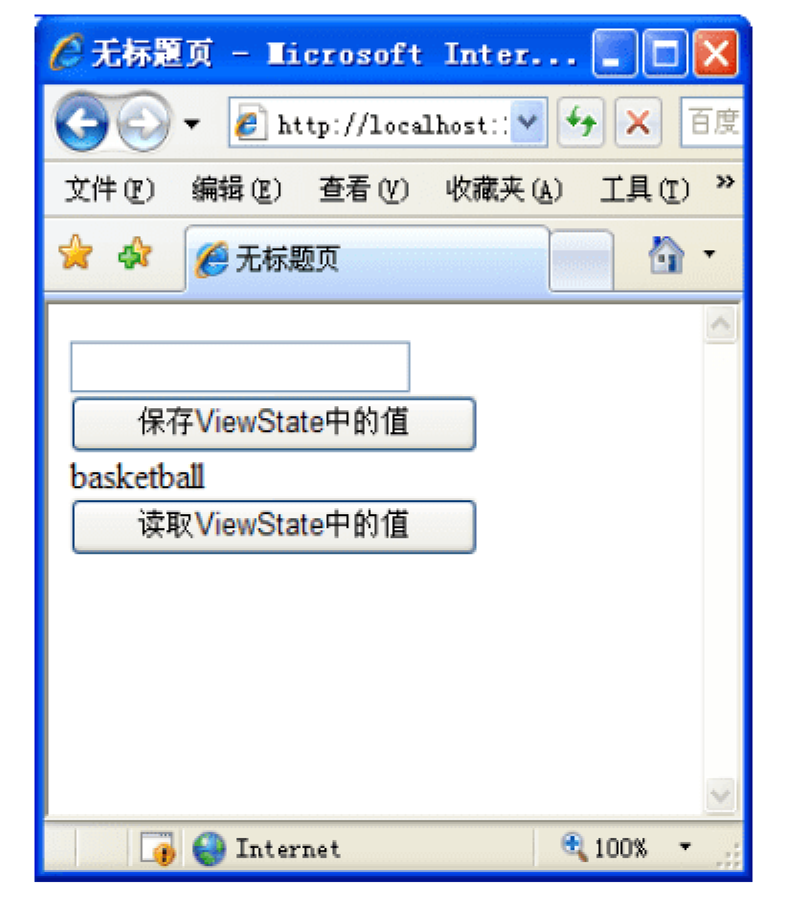


图 3-8　读取 ViewState 中的初值

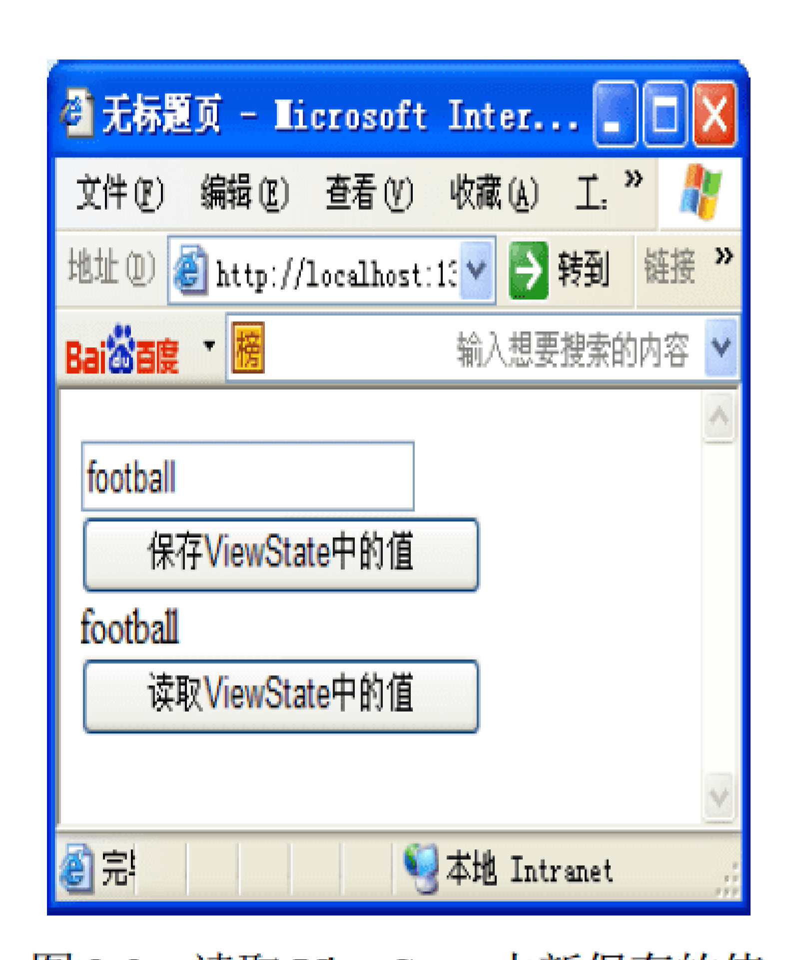


图 3-9　读取 ViewState 中新保存的值

3.3.7　Cookie 对象

Cookie 对象是比较常用的一种对象，可以用来保存一些常用的信息。

1. Cookie 对象简介

Cookie 俗称“小甜饼”，通常是用于存储少量的浏览者的信息，如浏览者的喜好、用户名、Email 地址等信息，以便于当浏览者再次登录网站时，不必再次填写这些信息。Cookie 其实只是一些小文本，将一些用户信息储存在客户端的机器中，它全部存储于 Windows 目录下的 Cookie 文件夹中，以便于在每次请求时被服务器在设定的时期内进行读取。Cookie 的储存大小是有限制的，一般浏览器会将其大小控制在 4096 个字节以内。

Cookie 与网站关联，而不是与特定的页面关联。因此，无论用户请求站点中的哪一个页面，浏览器和服务器都将交换 Cookie 信息。用户访问不同站点时，各个站点都可能会向用户的浏览器发送一个 Cookie，浏览器会分别存储所有 Cookie。

每次用户访问站点时，Web 应用程序都可以读取 Cookie 信息。当用户请求站点中的页面时，应用程序发送给该用户的不仅仅是一个页面，还有一个包含日期和时间的 Cookie，用户的浏览器在获得页面的同时也获得了该 Cookie，并将它存储在用户硬盘上的文件夹中。以后，如果该用户再次请求站点中的页面，当该用户输入 URL 时，浏览器便会在本地硬盘上查找与该 URL 关联的 Cookie。比如当用户登录某些网站的邮箱后，如果在 Cookie 中记录了

用户名这个信息，那么在 Cookie 信息失效以前，该用户在同一台计算机再次登录时就不需要提供用户名。

2. Cookie 的限制

大多数浏览器支持最大为 4096 字节的 Cookie。由于这限制了 Cookie 的大小，最好用 Cookie 来存储少量数据，或者存储用户 ID 之类的标识符。用户 ID 随后便可用于标识用户，以及从数据库或其他数据源中读取用户信息。

浏览器还限制站点可以在用户计算机上存储的 Cookie 的数量。大多数浏览器只允许每个站点存储 20 个 Cookie，如果试图存储更多的 Cookie，则存放最早的 Cookie 便会被覆盖掉。有些浏览器还会对它们将接受的来自所有站点的 Cookie 总数做出绝对限制，通常为 300 个。另外一个可能遇到的 Cookie 限制是用户可以将其浏览器设置为拒绝接受 Cookie。

3. 读取 Cookie

浏览器向服务器发出请求时，会随请求一起发送该服务器的 Cookie。在 ASP.NET 应用程序中，可以使用 HttpRequest 对象读取 Cookie，该对象可用作 Page 类的 Request 属性使用。HttpRequest 对象的结构与 HttpResponse 对象的结构基本相同，因此，可以从 HttpRequest 对象中读取 Cookie，并且读取方式与将 Cookie 写入 HttpResponse 对象的方式基本相同。

4. 修改和删除 Cookie

服务器不能直接修改 Cookie。如果要修改 Cookie，必须首先创建一个具有新值的新 Cookie。

例如：HttpCookie cookie=new HttpCookie("hello")

建立起一个名为 hello 的 HttpCookie 实例。

建立实例后，将给其赋值。在一个 Cookie 中可以储存一个值，也可以储存多个值。通过设置 Cookie 的 Value 属性值，可以在 Cookie 中储存一个值。代码为：

```
Cookie.Value="test"
```

通过 Cookie 的 Values 集合，可以在同一个 Cookie 中储存多个值。代码为：

```
HttpCookie cookie=new HttpCookie("test");
Cookie.Values.Add("Admin", "刘一");
Cookie.Values.Add("Member1", "李四");
Cookie.Values.Add("Member2", "王五");
Cookie.Values.Add("Member3", "张三");
```

Values 集合使用的 Add 方法中第一个参数为关键字(Key)，第二个参数是设置的值(Value)。

删除 Cookie 是修改 Cookie 的一种形式。由于 Cookie 在用户计算机中，因此无法将其直接移除。但是，可以让浏览器来删除 Cookie，该技术是创建一个与要删除的 Cookie 同名的新 Cookie，并将该 Cookie 的到期日期设置为过去的某个日期，当浏览器检查 Cookie 的到期

日期时，浏览器便会丢弃这个现已过期的 Cookie。

5. 确定浏览器是否接受 Cookie

用户可将其浏览器设置为拒绝接受 Cookie。虽然不能向客户端写入 Cookie 信息，但是不会引发任何错误。同样，浏览器也不向服务器发送有关其当前 Cookie 设置的任何信息。确定 Cookie 是否被接受的一种方法时尝试编写一个 Cookie，然后再尝试读取该 Cookie。如果无法读取已编写的 Cookie，则可以假定浏览器不接受 Cookie。

【例 3-6】通过一个例子来说明如何编写 Cookie 与读取 Cookie。

(1) 在 webside 网站中添加一个名为 Cookie.aspx 的网页。

(2) 在 Cookie.aspx.cs 页面中添加如下代码。

```
protected void ButtonWrite_Click(object sender, EventArgs e)
 {
      Response.Cookies["userName"].Value = "zhengzhou University";
      Response.Cookies["userName"].Expires = DateTime.Now.AddDays(1);
      this.Label1.Text = "写入完毕！ ";
 }
 protected void ButtonRead_Click(object sender, EventArgs e)
 {
      if (Request.Cookies["userName"] != null)
      {
           Label1.Text = Server.HtmlEncode(Request.Cookies["userName"].Value);
      }
}
```

(3) 程序运行效果如图 3-10 和图 3-11 所示。

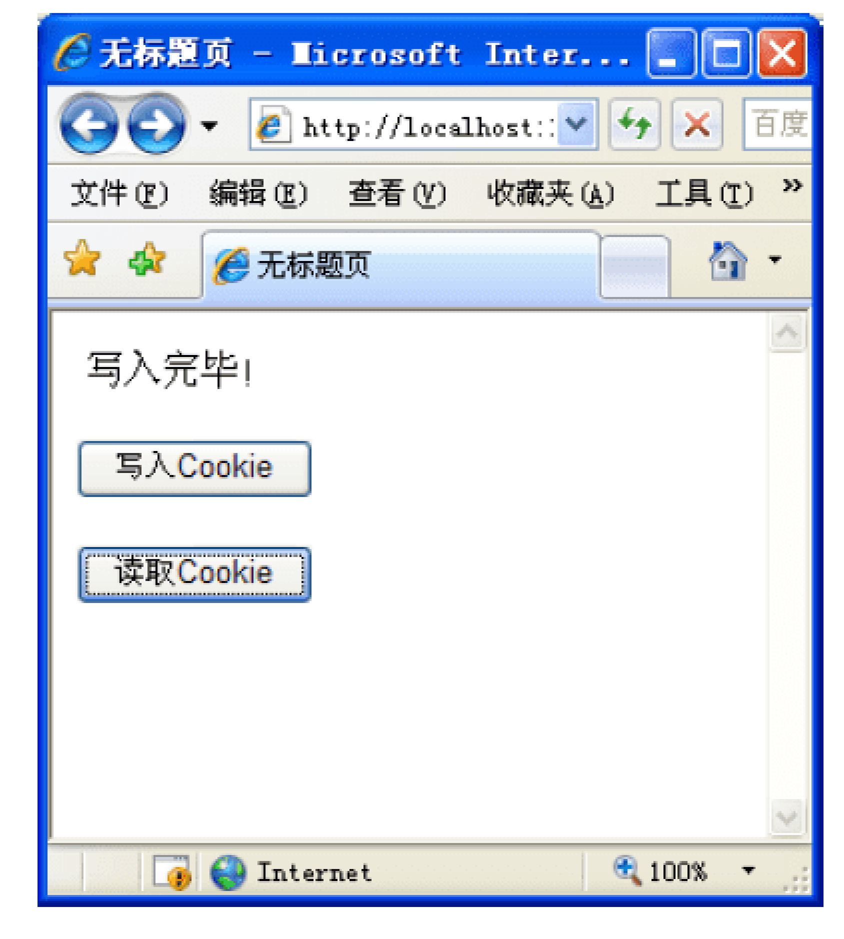


图 3-10　cookie.aspx 写 cookie 运行效果

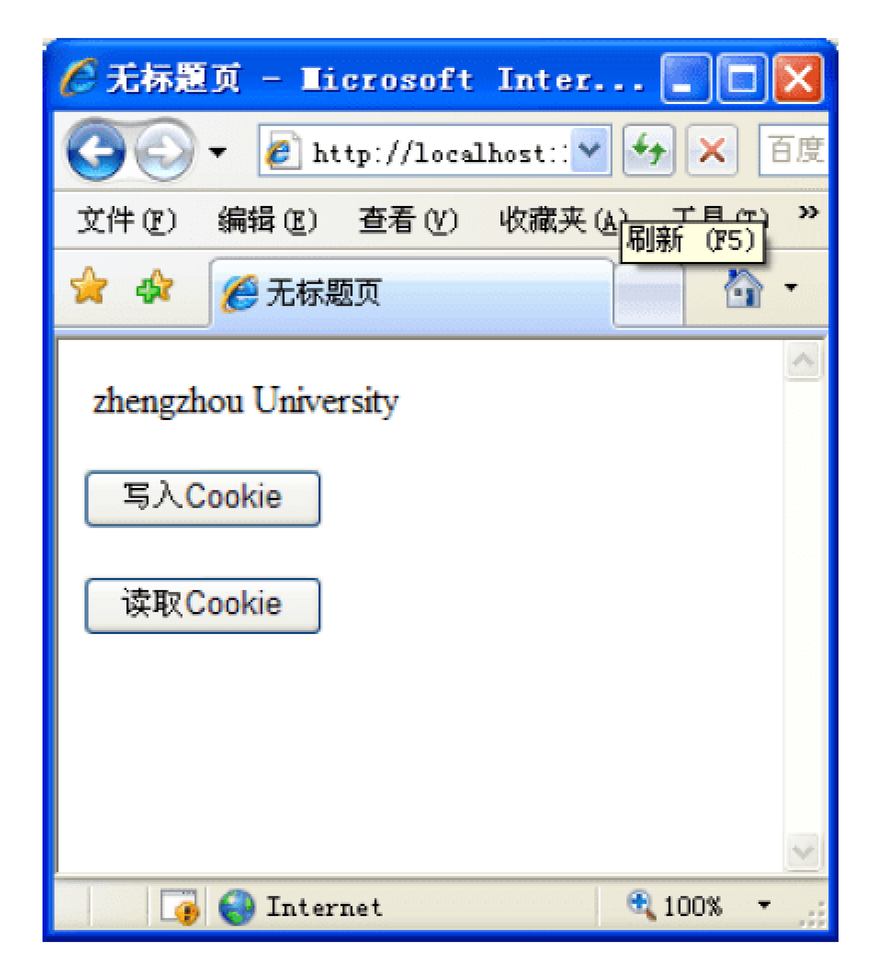


图 3-11　cookie.aspx 读 cookie 运行效果

3.4　ASP.NET 配置管理

使用 ASP.NET 配置系统的功能，可以配置整个服务器上的所有 ASP.NET 应用程序、单个 ASP.NET 应用程序和各个页面或应用程序子目录，也可以配置各种具体的功能，如身份验证模式、页缓存、编译器选项、自定义错误、调试和跟踪选项等。

3.4.1　web.config 文件介绍

ASP.NET 提供了一个丰富而可行的配置系统，以帮助管理人员轻松快速地建立自己的 Web 应用环境。

Web 配置文件 web.config 是 Web 应用程序的数据设定文件，它是一份 XML 文件，内含 Web 应用程序相关设定的 XML 标记，可以用来简化 ASP.NET 应用程序的相关设定。该文件位于 Web 应用程序的任何目录中，统一命名为 web.config，它决定了所在目录及其子目录的配置信息，并且子目录下的配置信息覆盖其父目录的配置，即子目录如果没有 web.config 文件，就是继承父目录 web.config 文件的相关设定；如果子目录有 web.config 文件，就会覆盖父目录 web.config 文件的相关设定。在运行状态下，ASP.NET 会根据远程 URL 请求，把访问路径下的各个 web.config 配置文件叠加，产生一个唯一的配置集合。

举例来说，一个对 URL:http://localhost\website\ownconfig\test.aspx 的访问，ASP.NET 会根据以下顺序来决定最终的配置情况：

(1) .\Microsoft.NET\Framework\{version}\web.config (默认配置文件)

(2) .\webapp\web.config　　　　(应用的配置)

(3) .\webapp\ownconfig\web.config　　　　(自己的配置)

web.config 是 aspx 区别于 asp 的一个方面，用户可以用这个文件配置自己的很多信息。ASP.NET 允许配置内容可以和静态内容、动态页面和商业对象放置在同一应用的目录结构下。当管理人员需要安装新的 ASP.NET 应用时，只需要将应用目录复制到新的机器上即可。

ASP.NET 的配置系统具有以下优点：

- ASP.NET 的配置内容以纯文本方式保存，可以以任意标准的文本编辑器、XML 解析器和脚本语言解释、修改配置内容。
- ASP.NET 提供了扩展配置内容的架构，以支持第三方开发者配置自己的内容。
- ASP.NET 配置文件的更改被系统自动监控，无须管理人员手工干预。

3.4.2　配置文件的语法规则

(1) 标识

配置内容被置于 web.config 文件中的标记<configuration>和</configuration>之间。

格式：

```
<configuration>
配置内容
…
</configuration>
```

(2) 配置段

具体定义配置的内容，以供应用使用。Web 配置文件是一份 XML 文件，在 XML 标记的属性就是设定值，标记名称和属性值格式是字符串，第一个开头字母是小写，之后每一字头是大写，例如<appSetting>。Web 配置文件的范例如下所示。

以下例子定义了一个 httpmodules 配置段，设置了系统 http 相关的处理模块。

```
<configuration>
    <appSettings>
      <add key="dbType" value="Access Database"/>
    </appsettings>
    <connectionsStrings>
<add name="provider"
connectionString="Microsoft.Jet.OLEDB.4.0;"/>
<add name="database"
connectionString="/chart7/Exam.mdb"/>
</connectionsStrings>
     <system.web>
<sessionState cookieless="false" timeout="10"/>
<compilation defaultLanguage="C#" debug="true"/>
<globalization
           fileEncoding="gb2312"
 requestEncoding="gb2312"
culture= "zh-CN"/>
           <customErrors mode="RemoteOnly"/>
      </system.web>
</configuration>
```

可以看到，在这段配置信息当中这是一个基于 XML 格式的文件，根标记是<configuration>，所有的配置信息均被包括在<configuration>及</configuration>标签中间，其子标记<appSettings>、<connectionsStrings>和<system.web>是各设定区段。在<system.web>下的设定区段属于 ASP.NET 相关设定。

在 Web 配置文件的<appSettings>区段可以创建 ASP.NET 程序所需要的参数，每个<add>标记可以创建一个参数，属性 key 是参数名称，value 是参数值。ASP.NET 2.0 以后新增<connectionStrings>区段，可以指定数据库连接字符串，在<connectionStrings>标记的<add>子标记也可以创建连接字符串，属性 name 是名称，connectionStrings 是连接字符串的内容，表 3-11 列出了常用设定区段标记的说明。

表 3-11　常用设定区段标记说明

设 定 区 段	说　　明
<AnonymousIdentification>	控制 Web 应用程序的匿名用户
<Authentication>	设定 ASP.NET 的验证方式
<Authorization>	设定 ASP.NET 用户授权
<BrowserCaps>	设定浏览程序兼容组件 HttpBrowserCapabilities
<Compilation>	设定 ASP.NET 应用程序的编译方式
<CustomErrors>	设定 ASP.NET 应用程序的自动错误处理
<Globalizations>	关于 ASP.NET 应用程序的全球化设定，也就是本地化设定
<HttpHandlers>	设定 HTTP 处理是对应到 URL 请求的 HttpHandler 类
<HttpModules>	创建\删除或清除 ASP.NET 应用程序的 HTTP 模块
<HttpRuntime>	设定 ASP.NET HTTP 运行时的配置，这些配置决定如何处理对 ASP.NET 应用程序的请求
<MachineKey>	设定在使用窗体基础验证的 Cookie 数据时，用来加码和解码的金钥匙
<Membership>	设定 ASP.NET 的 Membership 机制
<Pages>	设定 ASP.NET 程序的相关设定，即 Page 指引命令的属性
<Profile>	设定个人化信息的 Profile 对象
<Roles>	设定 ASP.NET 的角色管理
<SessionState>	设定 ASP.NET 应用程序的 Session 状态 HttpModule
<SiteMap>	设定 ASP.NET 网站导航系统
<WebParts>	设定 ASP.NET 应用程序的网页组件
<WebServices>	设定 ASP.NET 的 Web 服务

3.4.3　Global.asax 文件介绍

作为网络应用程序，程序在执行之前有时需要初始化一些重要的变量，而且这些工作必须发生在所有程序执行之前，ASP.NET 的 Global.asax 文件便是为此目的而设计的。每个 ASP.NET 应用程序都可以有一个 Global.asax 文件。一旦将其放在适当的虚拟目录中 ASP.NET 就会把它识别出来并且会自动使用该文件。

另外，由于 Global.asax 在网络应用程序中的特殊地位，它被存放的位置也是固定的。必须存放在当前应用所在的虚拟目录的根目录下。如果放在虚拟目录的子目录中，Global.asax 文件将不会起任何作用。

ASP.NET 在应用程序中添加了“全局应用程序类”，也就是 Global.asax。该文件是应用程序用来保持应用程序级的事件、对象和变量的。一个 ASP.NET 应用程序只能有一个 Global.asax 文件，位于应用程序根目录下。

按照 VWD 模板添加的应用程序如下所示。

```
<%@ Application Language="C#"%>
```

```
<script runat="server">
    void Application_Start(object sender,EventArgs e)
    {
        //在应用程序启动时运行的代码
    }
        void Application_End(object sender, EventArgs e)
    {
        //在应用程序关闭时运行的代码
    }
     void Application_Error(object sender, EventArgs e)
    {
        //在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender, EventArgs e)
    {
        //在新会话启动时运行的代码
    }
</script>
```

在窗体页中，只能处理单个页面的事件，而在 Global.asax 文件中可以处理整个应用程序中的事件。除了上述代码模板中列举的事件以外，在 Global.asax 文件中还可以加入其他事件的处理函数。表 3-12 列出了可以在 Global.asax 中处理的事件。

表 3-12　Global.asax 中处理的事件

事　件	说　明
Application_Start	在应用程序接收到第一个请求时调用，通常在此函数中定义应用程序级变量或状态
Session_Start	类似与 Application_Start，不过是针对每个客户端第一次访问应用程序时调用
Application_BeginRequest	虽然在 VWD 的代码模板中没有该事件的处理，不过可以在 Global.asax 中添加。该事件是在每个请求到达服务器，并且在处理该请求前，会触发的事件
Application_AuthenticateRequest	每个请求都会触发该事件，并且可以在此函数中设置自定义的验证
Application_Error	在应用程序中抛出任何错误时都会触发该事件。通常在此函数中提供应用程序级的错误处理或者记录错误事件
Session_End	以进程内模式使用会话状态时，如果用户离开应用程序将会触发该事件
Application_End	应用程序关闭时触发该事件。该函数很少使用，因为 ASP.NET 可以很好地关闭和清除内存对象

与页面指令一样，Global.asax 文件也可以使用应用程序指令，这些指令都可以包含特定

于该指令的一个或多个属性/值对。下面列出了 ASP.NET 中支持的应用程序指令。

- @Application。定义 ASP.NET 应用程序编译器所使用的应用程序特定的属性。该指令只能在 Global.asax 文件中使用。
- @Import。显式将命名空间导入到应用程序中。
- @Assembly。在分析时将程序集链接到应用程序。

实验 3　ASP.NET 基础知识

1. 实验目的

通过实践练习，进一步理解各章节知识，了解 ASP.NET 页面的运行机制和配置文件管理方式，熟练掌握 ASP.NET 各种对象的使用方法。

2. 实验内容和要求

(1) 编写程序，实现让用户输入两个整数，并使用一个按钮来进行数据提交，使用 Request 对象来得到用户提交的数据，比较 Get 方法和 Post 方法的不同。

(2) 编写程序，利用 Session 保存用户输入的信息，并在页面上进行显示。

(3) 编写程序，使用 Cookie 记录用户名，并在页面上显示出来。

第4章　ASP.NET服务器控件

ASP.NET 服务器控件是 ASP.NET 网页上的对象，当客户端浏览器请求服务器端的网页时，这些控件对象将在服务器上运行并向客户端浏览器呈现 HTML 标记。使用 ASP.NET 服务器控件，可以大幅减少开发 Web 应用程序所需编写的代码量，提高开发效率和 Web 应用程序的性能。

本章首先介绍 ASP.NET 服务器控件的基本概念，然后详细介绍在 ASP.NET 网页中如何添加 ASP.NET 服务器控件，以及这些控件的使用方法。

本章的学习目标

- 了解 ASP.NET 服务器控件的定义
- 掌握如何向网页中添加 ASP.NET 服务器控件，如何配置控件的属性、行为和外观以及如何以编程方式使用这些控件

4.1　ASP.NET 服务器控件概述

在网页上经常看到填写信息用的文本框、单选按钮、复选框、下拉列表等元素，它们都是控件。控件是可重用的组件或对象，有自己的属性和方法，可以响应事件。

ASP.NET 服务器控件是服务器端 ASP.NET 网页上的对象，当用户通过浏览器请求 ASP.NET 网页时，这些控件将运行并把生成的标准的 HTML 文件发送给客户端浏览器来呈现。

网站部署在 Web 服务器上，人们可以通过浏览器来访问这个站点。当客户端请求一个静态的 HTML 页面时，服务器找到对应的文件直接将其发送给用户端浏览器；而在请求 ASP.NET 页面时(扩展名为.aspx 的页面)，服务器将在文件系统中找到并读取对应的页面，然后将页面中的服务器控件转换成浏览器可以读取的 HTML 标记和一些脚本代码，再将转换后的结果页面发送给用户。

在 ASP.NET 页面上，服务器控件表现为一个标记，例如<asp:textbox…/>。这些标记不是标准的 HTML 元素，因此如果它们出现在网页上，浏览器将无法理解它们，然而，当从 Web 服务器上请求一个 ASP.NET 页面时，这些标记都将动态地转换为 HTML 元素，因此浏览器只会接收到它能理解的 HTML 内容。

在创建.aspx 页面时，可以将任意的服务器控件放置到页面上，然而请求服务器上该页面

的浏览器将只会接收到 HTML 和 JavaScript 脚本代码，如图 4-1 所示。

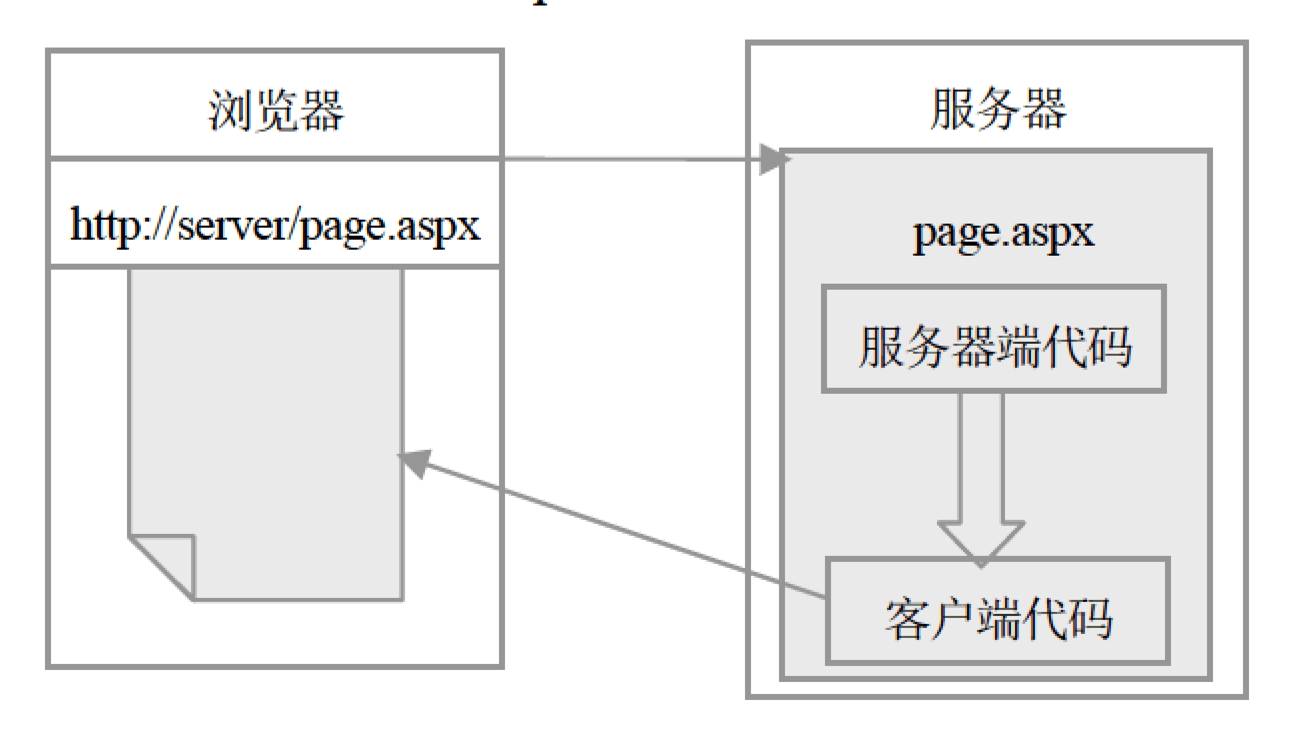


图 4-1　服务器端代码转换为客户端代码

记住，Web 浏览器无法理解 ASP.NET。Web 浏览器只理解 HTML，可能也理解 JavaScript——但它不能处理 ASP.NET 代码。服务器读取 ASP.NET 代码并进行处理，将所有 ASP.NET 特有的内容转换为 HTML 以及(如果浏览器支持的话)一些 JavaScript 代码，然后将最新生成的 HTML 发送回浏览器。

ASP.NET 服务器控件主要类别有 HTML 服务器控件、标准服务器控件、验证控件、用户控件、数据控件、导航控件等。

本章主要介绍 HTML 服务器控件、标准服务器控件、验证控件和用户控件。

4.2　HTML 元素和服务器控件

本节描述如何使用 VWD 2008 工具箱的 HTML 选项卡上的 ASP.NET 服务器控件。

4.2.1　HTML 元素

在 VWD 2008 中，从工具箱添加到页面上的 HTML 服务器控件只是已设置了某些属性的 HTML 元素，当然也可通过输入 HTML 标记在“源”视图中创建 HTML 元素。

默认情况下，ASP.NET 文件中的 HTML 元素作为文本进行处理，并且不能在服务器端代码中引用这些元素，只能在客户端通过 javascript 和 vbscript 等脚本语言来控制。

工具箱的 HTML 选项卡上提供了一些基于 HTML INPUT 元素的控件。本节通过下列两个控件介绍 HTML INPUT 元素的使用方法。

(1) Input (Button)控件，按钮控件：默认情况下是“INPUT type="button"元素”。

(2) Input (Text) 控件，文本框控件：默认情况下是“INPUT type="text"元素”。

Input(Text)元素创建允许用户在其中输入文本或密码的单行文本框，其主要属性有：

(1) Type=text/password，文本框的类型。

(2) MaxLength，文本框中最大的输入字节。

(3) Size，设定文本框的宽度。

(4) Value，设定文本框的值。

Input(Button)元素的主要功能是创建一个用来触发事件处理程序的按钮，通过使用onclick 属性来表明单击按钮可以触发的处理方法。其主要属性有：

(1) ID，此控件的编程名称。

(2) value，设置按钮中显示的文字。

【例 4-1】演示如何使用 Input (Text)元素和 Input (Button)元素。

(1) 运行 VWD 2008，新建名为 HTMLGroupControl 的 ASP.NET 网站。

(2) 重命名 default.apsx 为 HTMLInputElement.aspx，切换到【设计】视图，从【工具箱】的 HTML 组中拖动 3 个 Input(Text)控件和 1 个 Input(Button)控件到页面，页面设计效果如图 4-2 所示，再切换到【源】视图可看到如下所示的代码。

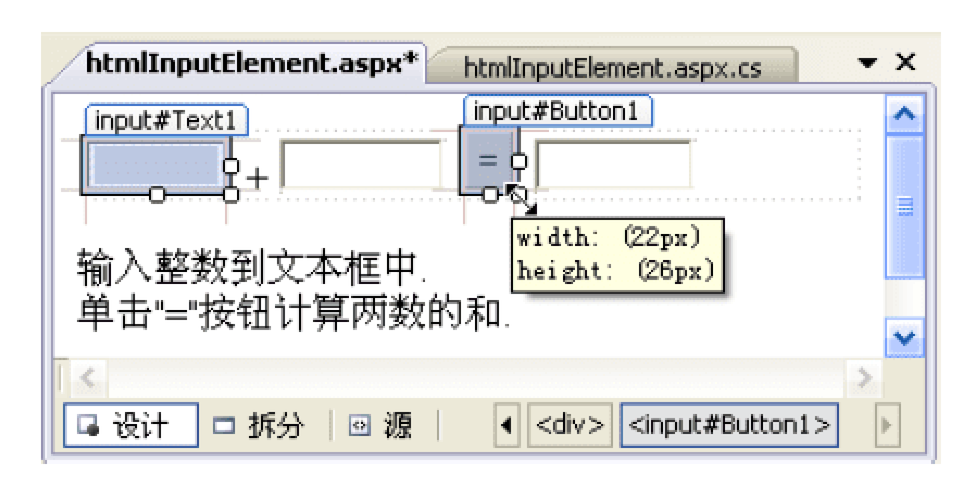


图 4-2　HTML image 控件

```
<input id="Text1" type="text" maxlength ="3"/>  + 
<input id="Text2" type="text" maxlength ="3"/> 
<input id="Button1" type="button" onclick="AddButton_Click();" value="=" /> 
<input id="Text3" type="text" maxlength="3" />
```

为文本框 Text1 和 Text2 分别添加属性 maxlength ="3"以控制输入字符的个数不超过 3 个。给按钮 Button1 控件添加属性 onclick="AddButton_Click();"，重置属性值 value="="。

(3) 切换到【源】视图，添加如下客户端脚本到<body>标记前。

```
<script type ="text/javascript" language ="javascript" >
    function AddButton_Click( )
    {
        alert('You clicked me.');
        var a= document.getElementById("Text1").value;
        var b=    document.getElementById("Text2").value;
        // alert(a);    alert(b);
        var answer=parseInt(a)+parseInt(b);
        document.getElementById("Text3").value=answer ;
        //alert(answer);
    }
</script>
```

(4) 按 Ctrl+F5，观看运行效果。单击=按钮时，将调用客户端脚本 AddButton_Click()。

4.2.2　HTML 服务器控件

在 VWD 2008 中，从工具箱添加到 ASP.NET 页面上的 HTML 服务器控件只是已设置了某些属性的 HTML 元素。

默认情况下，这些添加到 ASP.NET 文件中的 HTML 元素被视为传递给浏览器的标记，作为文本进行处理，不能在服务器端的代码中引用这些元素。若要使这些元素能以编程方式进行访问，可以通过添加 runat="server"属性表明应将 HTML 元素作为服务器控件进行处理。还可设置 HTML 元素的 id 属性，这样就可使用基于服务器的代码对其进行编程引用了。

添加了属性 runat="server"的 HTML 元素就转换为 HTML 服务器控件了。移除控件标记中的 runat="server"属性，HTML 服务器控件就转换为 HTML 元素了。

【例 4-2】演示如何使用工具箱的 Input(Button)服务器控件、Input(Text)服务器控件。

(1) 打开【例 4-1】所建的 HTMLGroupControl 网站。

(2) 新建名为 HTMLInputServerControl.aspx 的 Web 页面文件。从【工具箱】的 HTML 组中拖动 3 个 Input(Text)控件和 1 个 Input(Button)控件到页面，如图 4-3 所示。切换到【源】视图可看到如下所示的代码。

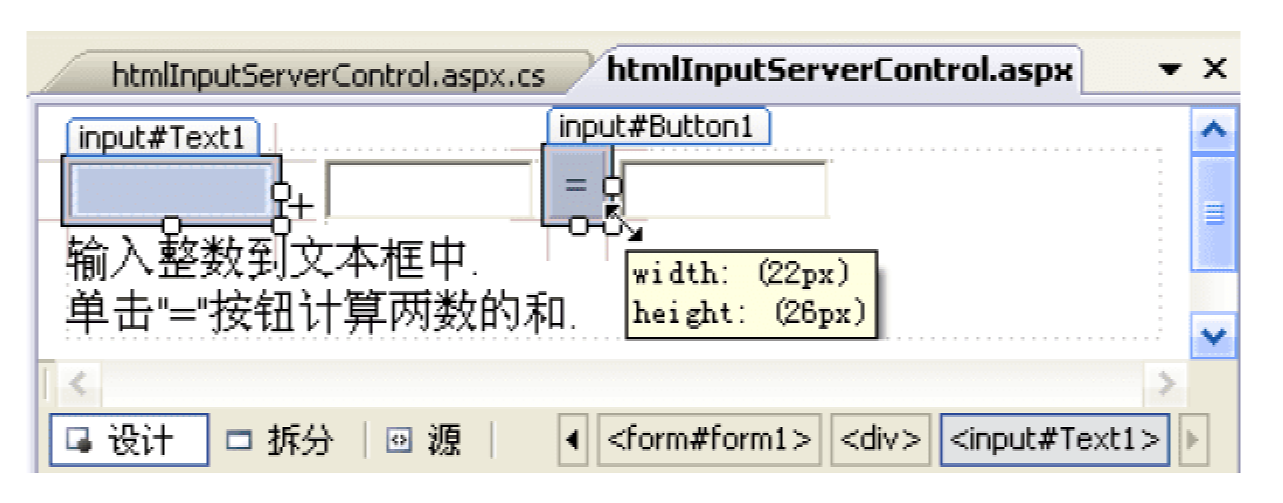


图 4-3　HTML input 控件设计视图

```
<input id="Text1" type="Text" size="2" maxlength="3" runat="server" /> +
<input id="Text2" type="Text" size="2" maxlength="3" runat="server" />
<input id="Button1" type="button" value="=" onserverclick="AddButton_Click" runat="server"/>
<input id="Text3" type="Text" size="2" maxlength="3" runat="server" />
```

为文本框 Text1 和 Text2 分别添加属性 maxlength ="3"以控制输入字符的个数不超过 3 个。给按钮 Button1 控件添加属性 onserverclick="AddButton_Click"，重置值 value="="。

给这 3 个文本框控件和这 1 个按钮控件都添加属性 runat="server"将其转换为“HTML 服务器控件”。

(3) 切换到 HTMLInputServerControl.aspx【源】视图，将下列服务器端脚本加到<body>标记前：

```
<script runat="server">
    protected void AddButton_Click(Object sender, EventArgs e)
    {
        int Answer;
        Answer = Convert.ToInt32(Text1.Value) + Convert.ToInt32(Text2.Value);
        Text3.Value = Answer.ToString();
```

```
    }
</script>
```

(4) 按 Ctrl+F5，在 IE 浏览器中观看运行效果。单击=按钮时，将调用服务器端脚本 AddButton_Click()。

HTML 控件由 HTML 元素衍生而来，由于 HTML 元素的属性只能静态地设置，一般在程序执行过程中不能被修改，很不灵活。为了弥补这一不足。ASP.NET 特地提供 HTML 控件，这种控件允许在程序中设置其属性，也允许在程序的执行过程中动态地读取及修改其属性，从而可以产生动态的网页。

【例 4-1】中的 HTML Input(Button)元素和【例 4-2】HTML 服务器控件的比较：

```
<input id="Button1" type="button" onclick="AddButton_Click();" value="=" />
<input id="Button1" type="button onserverclick="AddButton_Click" runat="server" " value="="/>
```

比较这两个语句，可以看出声明一个 HTML 服务器按钮控件和声明一个 HTML 元素有以下几点不同：

(1) 服务器按钮控件用 onserverclick 属性代替了 html 按钮的 onclick 属性。

(2) 服务器按钮控件多用了一个属性 runat＝"server"。

说明：

(1) ID 属性是标识服务器控件的唯一标志，通过它可以像引用一个对象一样来直接在服务器端引用服务器控件。

(2) 用 OnServerClick 属性代替 OnClick 属性是为了表明响应按钮的单击事件是在服务器端进行处理，而不是客户端。

(3) 多出的 runat＝"server"属性是说明该控件为服务器控件，这是区别 HTML 服务器控件和 html 元素的唯一方法。

4.3　标准服务器控件

ASP.NET 包含大量可在 ASP.NET 网页上使用的标准服务器控件。本节重点从 ASP.NET 服务器控件的工作原理和过程入手，介绍控件的常用功能和用法。

4.3.1　按钮控件

可使用 ASP.NET Button 服务器控件为用户提供向服务器发送网页的能力。该控件会在服务器代码中触发一个事件，可以处理该事件来响应回发。

ASP.NET 包括三种按钮控件，每种按钮控件在网页上显示的方式都不同，如表 4-1 所示。

表 4-1 按钮控件的比较

控　件	说　明
Button	显示一个标准命令按钮，该按钮呈现为一个 HTML input 元素
LinkButton	呈现为页面中的一个超链接。但是，它包含使窗体被发回服务器的客户端脚本(可以使用 HyperLink 服务器控件创建真实的超链接)
ImageButton	将图形呈现为按钮。这对于提供丰富的按钮外观非常有用。ImageButton 控件还提供有关图形内已单击位置的坐标信息

(1) 按钮事件

当用户单击任何 Button(按钮)服务器控件时，会将该页发送到服务器。这使得在基于服务器的代码中，网页被处理，任何挂起的事件被引发。这些按钮还可引发它们自己的 Click 事件，可以为这些事件编写事件处理程序。

(2) 按钮回发行为

当用户单击按钮控件时，该页回发到服务器。默认情况下，该页回发到其本身，在这里重新生成相同的页面并处理该页上控件的事件处理程序。

可以配置按钮以将当前页面回发到另一页面。这对于创建多页窗体可能非常有用。

默认情况下，Button 控件使用 HTML POST 操作提交页面。LinkButton 和 ImageButton 控件不能直接支持 HTML POST 操作。因此，使用这些按钮时，它们将客户端脚本添加到页面以允许控件以编程方式提交页面。(因此 LinkButton 和 ImageButton 控件要求在浏览器上启用客户端脚本。)

在某些情况下，可能希望 Button 控件也使用客户端脚本执行回发。这在希望以编程方式操作回发(如将回发附加到页面上的其他元素)时非常有用。可以将 Button 控件的 UseSubmitBehavior 属性设置为 true 以使 Button 控件使用基于客户端脚本的回发。

(3) 处理 Button 控件的客户端事件

Button 控件既可以引发服务器事件，也可以引发客户端事件。服务器事件在回发后发生，且这些事件在为页面编写的服务器端代码中处理。客户端事件在客户端脚本(通常为 ECMAScript(JavaScript))中处理，并在提交页面前引发。通过向 ASP.NET 按钮控件添加客户端事件，可以执行一些任务(如在提交页之前显示确认对话框以及可能取消提交)。

【例 4-3】演示 Button 控件、LinkButton 控件、ImageButton 控件的使用。

(1) 打开 VWD 2008，新建名为 ButtonWebServerControl 的 ASP.NET 网站。

(2) 新建页面 Poem.aspx 和 prose.aspx。在网站根目录下新建名为 Image 的文件夹，将选中的图片添加到 Image 文件夹中。

(3) 切换到 default.aspx 的【设计】视图，在【工具箱】中双击相应的控件，添加 1 个 HyperLink 控件、1 个 LinkButton 控件、1 个 Button 控件、1 个 ImageButton 控件和 1 个 Label 控件。如图 4-4 所示。

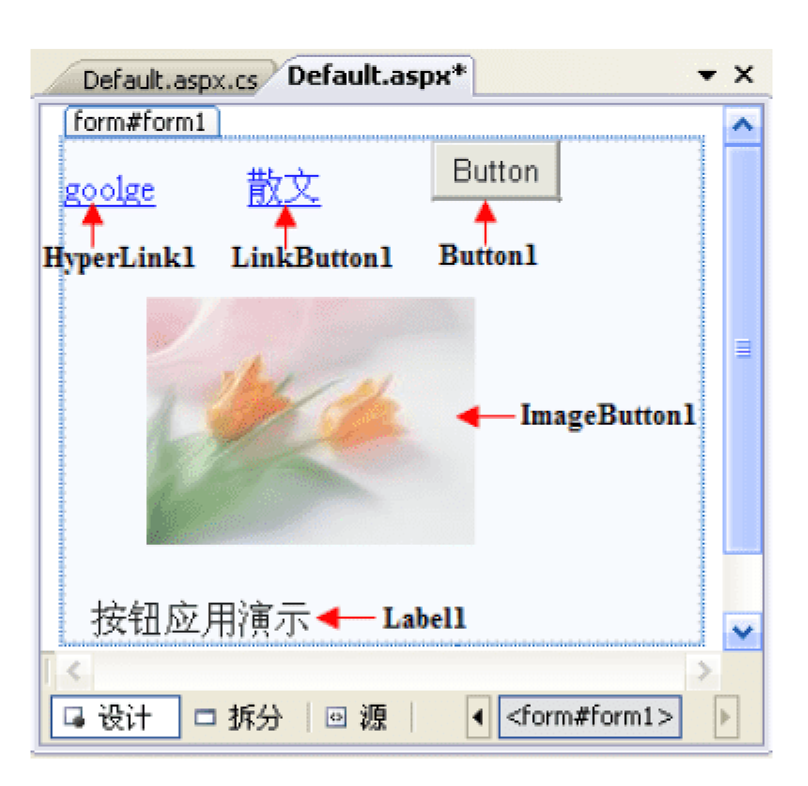


图 4-4　创建按钮控件

(4) 在【设计】视图中，单击 HyperLink1 控件，在其属性窗口设置 Text 属性值为 Google，NavigateUrl 属性值分别为 http://www.google.cn。

同样，设置 LinkButton1 控件的 Text 属性值为“散文”，PostBackUrl 属性值为~/prose.aspx；ImageButton1 控件的 ImageUrl 属性值为~/Image/flower.jpg，PostBackUrl 属性值为~/flower.aspx，如下标记代码所示：

```
<asp:HyperLink ID="HyperLink1" runat="server"
    NavigateUrl="http://www.google.cn">goolge</asp:HyperLink>
<asp:LinkButton ID="LinkButton1" runat="server" PostBackUrl = "~/prose.aspx"
    onclick="LinkButton1_Click">散文</asp:LinkButton>
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~/Image/flower.jpg"
    onclick="ImageButton1_Click" PostBackUrl="~/flower.aspx"
    AlternateText="this is a ImageButton." />
<asp:Label ID="Label1" runat="server" Text="按钮应用演示"></asp:Label>
```

(5) 在【设计】视图中，双击 Button1 控件，为 Button1 的 Click 事件添加如下所示的处理程序。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "哈哈，您单击的是 Button 按钮！ ";
}
```

(6) 按 Ctrl+F5，观看显示效果。

4.3.2　TextBox(文本框)控件

TextBox 服务器控件是让用户向 ASP.NET 网页输入文本的控件。默认情况下，该控件的 TextMode 属性设置为 TextBoxMode.SingleLine，这将显示一个单行文本框。但可以将 TextMode 属性设置为 TextBoxMode.MultiLine，以显示多行文本框(该文本框将作为 textarea 元素呈现)。也可以将 TextMode 属性更改为 TextBoxMode.Password，以显示屏蔽用户输入的文本框。通过使用 Text 属性可以获得 TextBox 控件中显示的文本。

安全说明：将 TextMode 属性设置为 TextBoxMode.Password 可有助于确保在输入密码时其他人无法看到。但是，输入到文本框中的文本没有以任何方式进行加密，为了提高安全性，在发送其中带有密码的页时，可以使用安全套接字层(SSL)和加密。

【例 4-4】演示文本框 TextBox 控件、标签 Label 控件、按钮 Button 控件的使用。

(1) 打开 VWD 2008，新建名为 TextBoxWebServerControl 的 ASP.NET 网站。

(2) 将 default.aspx 重命名为 TestButton.aspx。

(3) 切换到如图 4-5 所示的【设计】视图，添加名为 Label1、Label2 的 Label 控件，名为 TextBox1、TextBox2 的 TextBox 控件，名为 Button1 的 Button 控件。

(4) 单击 Label1 控件，在其属性窗口设置 Text 属性值为“原始内容：”；同样，设置 Label2 控件的 Text 属性值为“克隆内容：”，Button1 控件的 Text 属性值为“克隆”。如图 4-5 所示。

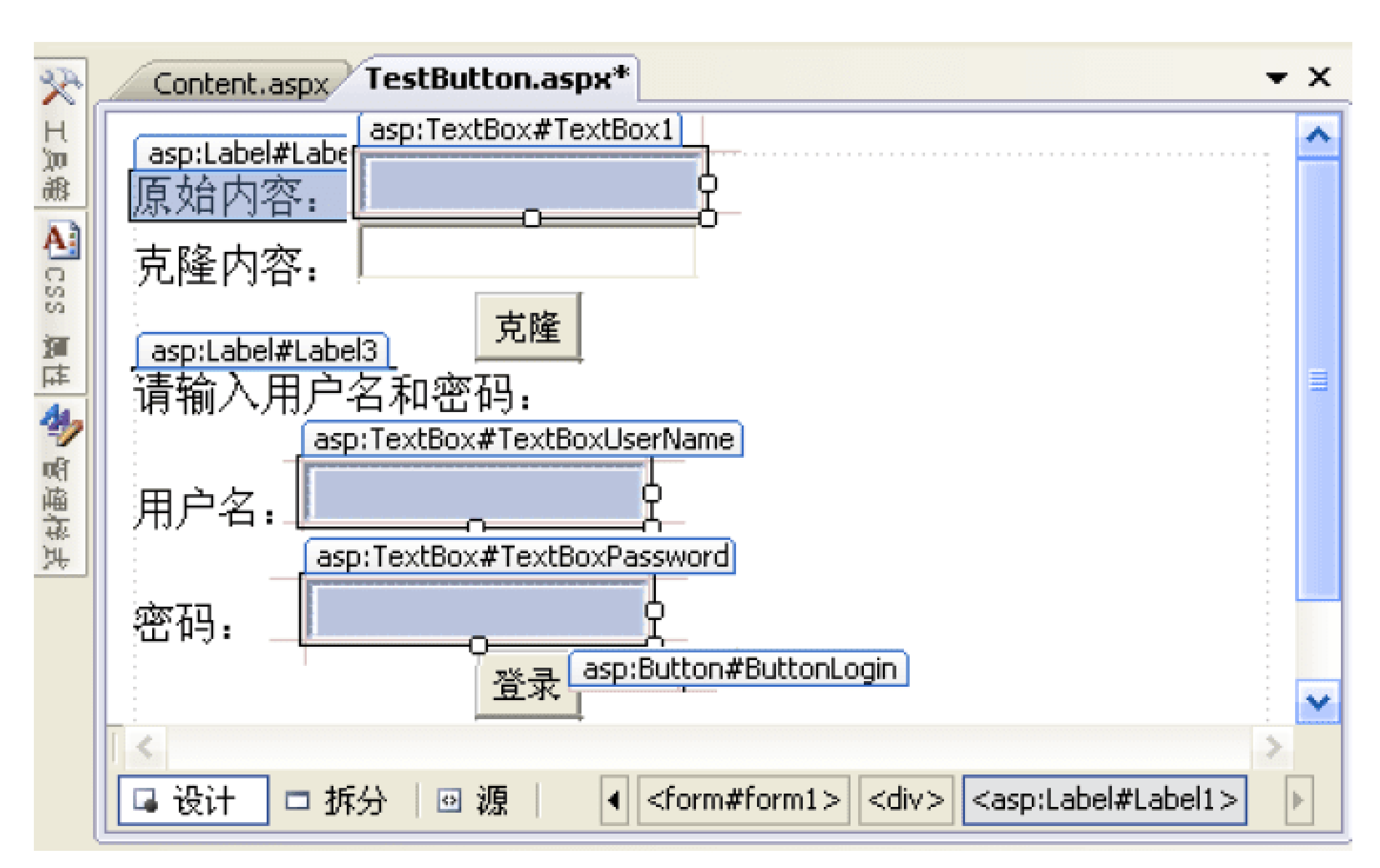


图 4-5　创建控件

(5) 在【设计】视图中，双击 Button1 控件，切换到 TestButton.aspx.cs 视图，为 Button1 控件的 Click 事件添加如下所示的处理程序。

```
protected void Button1_Click(object sender, System.EventArgs e)
{
    //读取 TextBox1 控件的 Text 属性的值，并使用该值来设置 TextBox2 控件的 Text 属性的值。
    TextBox2.Text = Server.HTMLEncode(TextBox1.Text);
}
```

(6) 按 Ctrl+F5，观看显示效果。

(7) 在【设计】视图中，分别单击 TextBox1、TextBox2 控件，在其属性窗口设置 TextMode(文本框的行为模式)属性值为 MultiLine。按 Ctrl+F5，观看显示效果。

(8) 按图 4-5 所示，添加 3 个 Label 控件，2 个 TextBox 控件，1 个 Button 控件，在属性窗口分别如图 4-5 所示设置 Text；设置 TextBox4 控件的 TextMode 属性值为 Password。

(9) 在【设计】视图中，双击【登录】按钮控件，切换到 TestButton.aspx.cs 视图，为 ButtonLogin 控件的 Click 事件的处理方法添加如下所示的代码。

```
Protected void ButtonLogin_Click(object sender, EventArgs e)
{
    if (TextBoxUserName.Text ==" " || TextBoxPassword.Text == "")
    {
        string scriptString = "alert("信息不全！请重新填写！"); ";
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(),"warning", scriptString, true);
    }
    else
    {
        Response.Redirect(string.Format("Content.aspx?user={0}&password={1}",
        TextBoxUserName.Text, TextBoxPassword.Text));
        //Response.Redirect("Content.aspx");
//Response.Redirect("Content.aspx?user="+TextBoxUser.Text+"&password="+TextBoxPassword.Text);
    }
}
```

在这段代码中，当单击 ButtonLogin 控件后，首先检查 TextBoxUserName 控件和 TextBoxPassword 控件中的内容是否为空，若为空，则弹出对话框，提示用户重新输入信息；若不空，则将控件中的数据作为参数传给 Content.aspx 页面。

(10) 在 Content.aspx 页面的【设计】视图下，双击 Web 窗体，在 Page_Load 事件中添加以下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    string scriptContent = Request.QueryString["user"].ToString() + "，欢迎您！注意记好您的密码：
       " + Request.QueryString["password"].ToString();
    Response.Write(scriptContent);
    string scriptString = "alert('"+scriptContent+"');";
    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", scriptString, true);
}
```

(11) 按 Ctrl+F5，观看显示效果。

4.3.3　ListBox 控件和 DropDownList 控件

ListBox 服务器控件使用户能够从预定义的列表中选择一项或多项。

DropDownList 服务器控件使用户可以从预定义的下拉列表中选择单个项，它与 ListBox 服务器控件的不同之处在于，其选项列表在用户单击下拉按钮之前一直保持隐藏状态。此外，DropDownList 控件不支持多重选择模式。

DropDownList 控件的某个选项被选中时，该控件将引发 SelectedIndexChanged 事件。默认情况下，此事件不会导致向服务器发送页，但可通过将 AutoPostBack 属性设置为 true，强制该控件立即发送。

如果将 ListBox 控件的属性 SelectionMode 的值从 Single 改为 Multiple，将允许进行多重选择，用户可以在按住 Ctrl 或 Shift 键的同时，单击以选择多个项。

【例 4-5】演示 ListBox 控件和 DropDownList 控件的使用。

(1) 运行 VWD 2008，新建名为 ListBoxDropDownList 的 ASP.NET 网站。

(2) 切换到 default.aspx 页【设计】视图，从【工具箱】中拖动 1 个 DropDownList 控件、1 个 ListBox 控件和 1 个 Label 控件到页面，如图 4-6 所示。

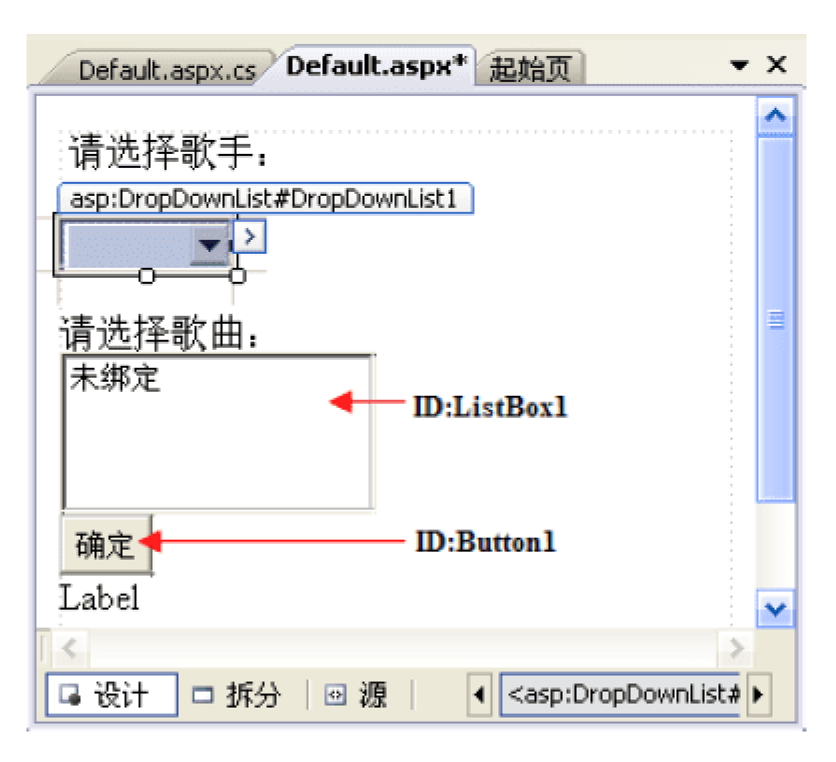


图 4-6　新建数据库

(3) 选择 DropDownList1 控件，单击右上角的【智能标记】，选择【启用 AutoPostBack】，再单击【编辑项…】，弹出如图 4-7 所示的对话框，单击【添加】按钮，添加 4 个成员并设置相应的 Text 属性值。



图 4-7　ListItem 集合编辑器

(4) 双击 DropDownList1 控件，添加其 SelectedIndexChanged 事件，代码如下：

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (DropDownList1 .SelectedItem .Text.Trim () )
    {
        case "刘欢":
            ListBox1.Items.Clear();
            ListBox1.Items.Add("从头再来");
            ListBox1.Items.Add("少年壮志不言愁");
            ListBox1.Items.Add("笑傲江湖");
            break;
```

```
                case "腾格尔":
                    ListBox1.Items.Clear();
                    ListBox1.Items.Add("天堂");
                    ListBox1.Items.Add("母亲河的泪水");
                    ListBox1.Items.Add("可爱的一朵玫瑰花");
                    break;
                case "蒋大伟":
                    ListBox1.Items.Clear();
                    ListBox1.Items.Add("桃花依旧笑春风");
                    ListBox1.Items.Add("在那桃花盛开的地方");
                    ListBox1.Items.Add("骏马奔驰保边疆");
                    break;
            }
        }
```

(5) 双击 Button1 控件，添加其 Click 事件，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    string msg = "您选择的歌曲是：";
    foreach (ListItem li in ListBox1.Items)
    {
        if (li.Selected == true)
        {
            msg += "<br />" + li.Text ;
        }
    }
    Label1.Text = msg;
}
```

(6) 选中 ListBox1 控件，在【属性】窗口设置属性 SelectionMode 的值为 Multiple。

(7) 按 Ctrl+F5，分别选择不同的选项，观看显示效果。

4.3.4　CheckBox(复选框)和 CheckBoxList(复选框列表)控件

CheckBox 控件和 CheckBoxList 控件分别用于向用户提供选项和选项列表。CheckBox 控件适合用在选项不多且比较固定的情况，当选项较多或需在运行时动态决定有哪些选项时，使用 CheckBoxList 控件比较方便。

【例 4-6】演示复选框控件 CheckBox 和复选框列表控件 CheckBoxlist 的使用。

(1) 运行 VWD 2008，新建名为 CheckBoxListWebServerControl 的网站。

(2) 切换到 Default.aspx 页的【设计】视图。

(3) 输入静态文本“CheckBoxList 控件演示程序”，在【格式】工具栏中使用【块格式】下拉列表将该文本的格式设置为“标题 1”。

(4) 添加 CheckBoxList1、Button1、CheckBox1、CheckBox2 和 Label1 这 5 个控件。如图

4-8 所示。



图 4-8　添加控件

(5) 在【设计】视图中，单击 CheckBoxList1 控件右上角的【智能标记】，切换到【CheckBoxList 任务】对话框，不选择【启用 AutoPostBack】，单击【编辑项…】，出现【ListItem 集合编辑器】，单击【添加】按钮，添加成员 ListItem0～ListItem5，分别将其属性 Text 值设置为 Visual Basic、C#、C++、Jscript、XMAL、Java。其 HTML 代码如下：

```
<asp:CheckBoxList ID=”CheckBoxList1” runat=”server”   RepeatDirection=”Horizontal”
          onselectedindexchanged=”CheckBoxList1_SelectedIndexChanged”>
    <asp:ListItem>Visual Basic</asp:ListItem>
    <asp:ListItem>c#</asp:ListItem>
    <asp:ListItem>C++</asp:ListItem>
    <asp:ListItem>Jscript</asp:ListItem>
    <asp:ListItem>XAML</asp:ListItem>
    <asp:ListItem>Java</asp:ListItem>
</asp:CheckBoxList>
```

(6) 设置 Button1 控件的 Text 属性值为“确定”，CheckBox1 控件的 Text 属性值为“改变显示风格”,CheckBox2 控件的 Text 属性值为“改变字体颜色”。如图 4-8 所示。

(7) 在【设计】视图中，分别双击 CheckBox1、CheckBox2 和 Button1 控件，切换到代码视图，添加如下所示的代码。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string str = "选择结果：";
    Label1.Text = "";
    for (int i = 0; i < CheckBoxList1.Items.Count; i++)
    {
        if (CheckBoxList1.Items[i].Selected)
        {
            //Label1.Text += CheckBoxList1.Items[i].Text + " <br/> ";
            str += CheckBoxList1.Items[i].Text + "、";
        }
    }
    if (str.EndsWith("、") == true) str = str.Substring(0, str.Length - 1);
```

```
        Label1.Text = str;
        if (str == "选择结果：")
        {
            string scriptString = "alert('请作出选择！');";
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning!",
                scriptString, true);
        }
        else
        {
            Label1.Visible = true;
            Label1.Text = str;
        }
    }
    protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
    {
        this.CheckBoxList1.BackColor =
            CheckBox1.Checked ? System.Drawing.Color.Beige : System.Drawing.Color.Azure;
        CheckBoxList1.RepeatDirection =
             CheckBox1.Checked ? RepeatDirection.Horizontal : RepeatDirection.Vertical;
    }
    protected void CheckBox2_CheckedChanged(object sender, EventArgs e)
    {
        if (CheckBox2.Checked)
        {
        this.CheckBoxList1.ForeColor = System.Drawing.Color.Red;
            Label1.ForeColor = System.Drawing.Color.Red;
        }
        else
        {
            this.CheckBoxList1.ForeColor = System.Drawing.Color.Black;
            Label1.ForeColor = System.Drawing.Color.Black;
        }
    }
```

(8) 按 Ctrl+F5，观看显示效果。

4.3.5　Image 和 ImageMap 控件

ASP.NET 3.5 包含两个图形控件，一个是 Image 控件，一个是 ImageMap 控件。

(1) Image 服务器控件使用户可以在 ASP.NET 网页上显示图像，并用自己的代码管理这些图像。可以在设计时或运行时以编程方式为 Image 对象指定图形文件。还可以将控件的 ImageUrl 属性绑定到一个数据源，以根据数据库信息显示图形。

与大多数其他服务器控件不同，Image 控件不支持任何事件。例如，Image 控件不响应鼠标单击事件。实际上，可以通过使用 ImageMap 或 ImageButton 控件来创建交互式图像。

(2) Imagemap 说白了就是一种图形，包括许多不同部分，将鼠标指针指在图形的各个部分，点一下鼠标左键，就可以进入另一个超级链接的页面。

Imagemap 不一定要真的是 map，可以是任何图形。比如一张脸，将鼠标点在眼睛上就出现关于眼睛的页面，点在鼻子上就出现鼻子的页面(很适合做成整容外科的网页吧！)，也是一种 ImageMap。

ImageMap 控件由两个元素组成。第一个是图像，它可以是任何标准 Web 图形格式的图形，如.gif、.jpg 或.png 文件。

第二个元素是 HotSpot(作用点)控件的集合。每个作用点控件都是一个类型为 CircleHotSpot、RectangleHotSpot 或 PolygonHotSpot 的不同项。对于每个作用点控件，都要定义用于指定该作用点的位置和大小的坐标。例如，如果创建一个 CircleHotSpot 控件，则需要定义圆心的 x 和 y 坐标以及圆的半径。

响应用户单击：每一个作用点都可以是一个单独的超链接或回发事件。可以指定用户单击作用点时发生的事件，可以将每个作用点配置为可以转到为该作用点提供的 URL 的超链接。或者，也可以将控件配置为在用户单击某个作用点时执行回发，并可为每个作用点提供一个惟一值。回发会引发 ImageMap 控件的 Click 事件。在事件处理程序中，可以读取分配给每个作用点的惟一值。

【例 4-7】演示 Image 控件和 ImageMap 控件的使用。

(1) 运行 VWD 2008，新建一个名为 ImageMapWebServerControl 的 ASP.NET 网站。

在网站的的根目录下新建名为 shared 的文件夹，再在 shared 文件夹中新建名为 images 的文件夹，将准备好的图片复制到 images 文件夹中。

(2) 切换到 Default.aspx 页的【设计】视图。

(3) 输入静态文本“ImageMap 控件导航示例”、“单击某个行星图片可看其更大的图片”，在【格式】工具栏中使用【块格式】下拉列表将该文本的格式设置为“标题 1”。

(4) 如图 4-9 所示，添加 ImageMap1、Image1 和 Image2 这 3 个控件。

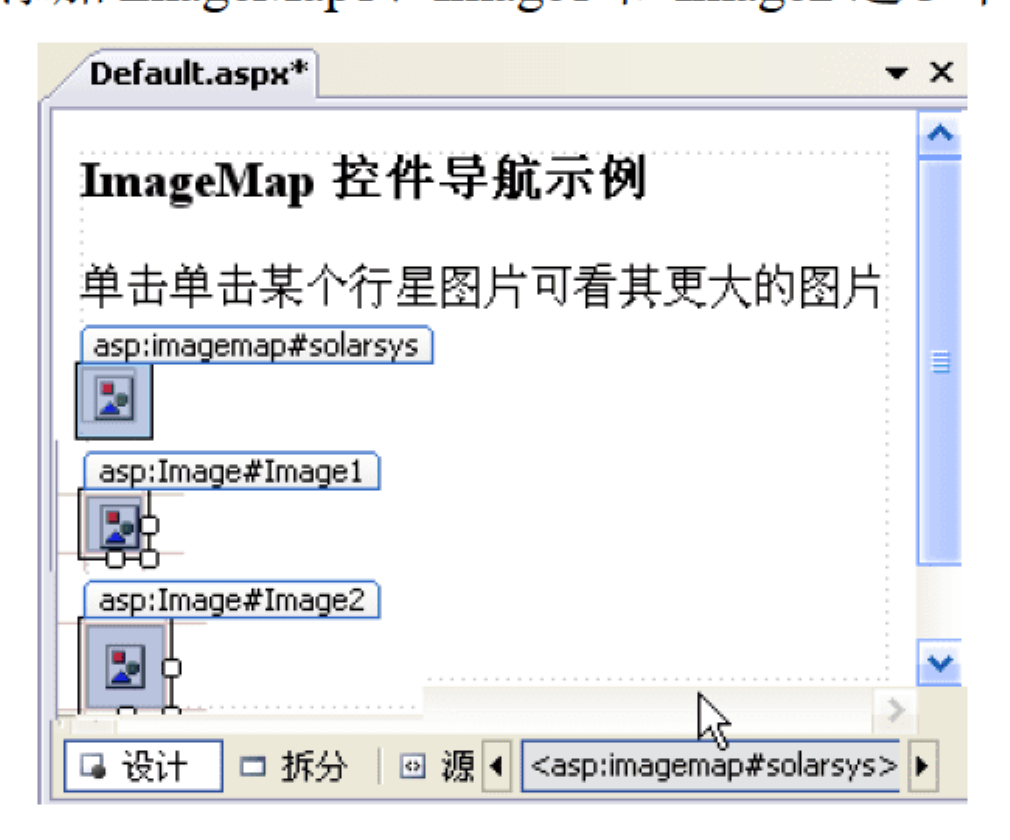


图 4-9　添加 ImageMap 控件和 Image 控件

(5) 在【设计】视图中，单击 ImageMap 控件，在其【属性】窗口，设置属性值为：id="solarsys"、imageurl="~/shared/images/solarsys.gif"、alternatetext="Solar System(太阳系) "，再单击【HotSpots】(作用点)属性框来显示【HotSpots 集合编辑器】对话框，单击【添加】按

钮旁边的箭头按钮来选择要创建的作用点类型(CircleHotSpot、RectangleHotSpot、PolygonHotSpot)，如图 4-10 所示，添加 10 个成员，并根据如下所示的语法设置其相应属性。



图 4-10　为 ImageMap 控件添加 HotSpot 热区

ImageMap 控件的 HTML 语法如下：

```
<asp:imagemap id="solarsys"     imageurl="~/shared/images/solarsys.gif" width="0px"
   height="0px" alternatetext="Solar System(太阳系) " runat="Server" onclick="solarsys_Click">
  <asp:rectanglehotspot alternatetext="Sun(太阳) " left="0" top="0" right="76" bottom="122"
    hotspotmode="Navigate" navigateurl="~/shared/images/sun.gif" target="_blank" />
  <asp:circlehotspot navigateurl="~/shared/images/mercury.gif" x="90" y="61" radius="15"
    hotspotmode="Navigate"   alternatetext="Mercury(水星) "   target="_blank" />
  <asp:circlehotspot   navigateurl="~/shared/images/venus.gif" x="124" y="61" radius="18"
    hotspotmode="Navigate"   alternatetext="Venus(金星) "    target="_blank" />
  <asp:circlehotspot   navigateurl="~/shared/images/earth.gif" x="163" y="61" radius="23"
    hotspotmode="Navigate"   alternatetext="Earth(地球) "    target="_blank" />
  <asp:circlehotspot     navigateurl="~/shared/images/mars.gif" x="201" y="61" radius="14"
      hotspotmode="Navigate" alternatetext="Mars(火星) "   target="_blank" />
  <asp:circlehotspot navigateurl="~/shared/images/jupiter.gif"   x="256" y="61" radius="40"
      hotspotmode="Navigate"   alternatetext="Jupiter(木星) " target="_blank" />
  <asp:circlehotspot   navigateurl="~/shared/images/saturn.gif" x="328" y="61" radius="35"
      hotspotmode="Navigate" alternatetext="Saturn(土星) " target="_blank" />
  <asp:circlehotspot navigateurl="~/shared/images/uranus.gif"    x="392" y="61" radius="30"
      hotspotmode="Navigate" alternatetext="Uranus(天王星) " target="_blank" />
  <asp:circlehotspot   navigateurl="~/shared/images/neptune.gif" x="443" y="61" radius="24"
      hotspotmode="Navigate" alternatetext="Neptune(海王星) " target="_blank" />
  <asp:circlehotspot   navigateurl="~/shared/images/pluto.gif" x="480" y="61" radius="13"
      hotspotmode="Navigate" alternatetext="Pluto(冥王星) " target="_blank" />
</asp:imagemap>
```

提示：

HotSpots(作用点)的坐标以像素为单位，在 Windows 的【画图】程序中打开，任选一种

画图工具，移动鼠标，在下部状态栏中可看到当前鼠标所在处的像素点的坐标，如图 4-11 所示。

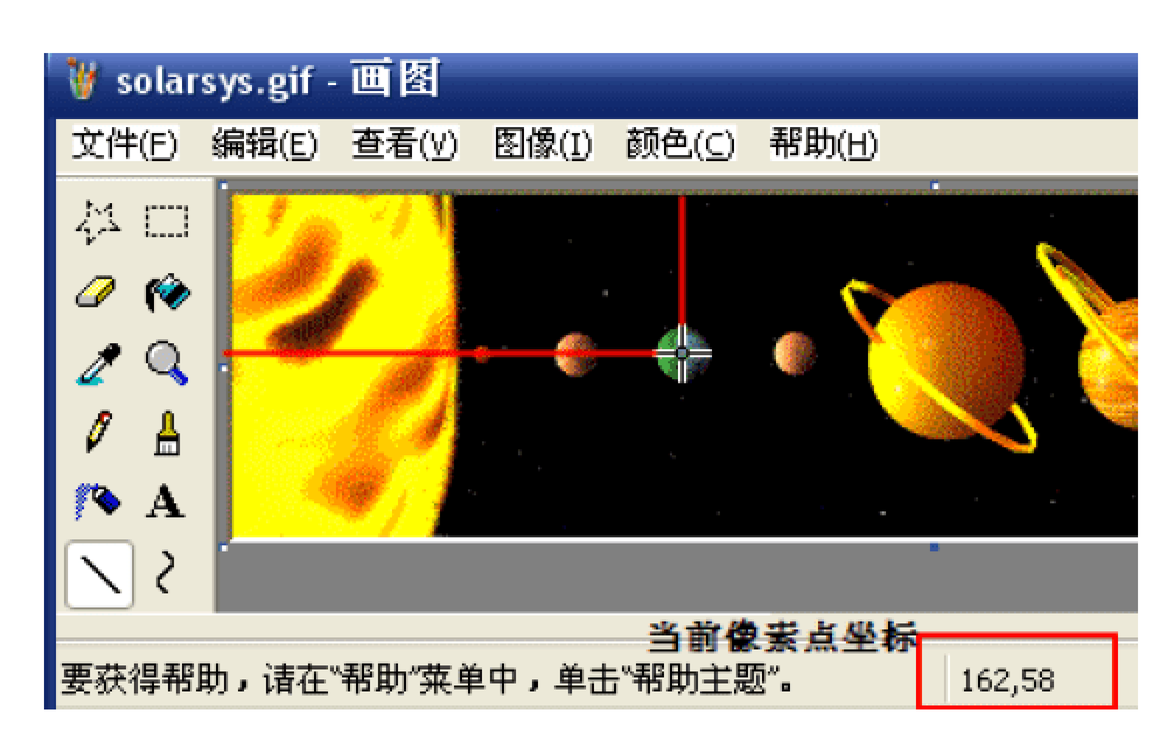


图 4-11　利用【画图】程序寻找图像当前光标处的像素点坐标

(6) Image1 控件的属性设置为：ImageUrl="~/shared/images/水果 1.jpg"，Image2 控件的属性设置为：ImageUrl="~/shared/images/水果 2.jpg"。

(7) 按 Ctrl+F5，观看显示效果。

4.3.6　RadioButton 和 RadioButtonList 服务器控件

在向 ASP.NET 网页添加单选按钮时，可以使用两种服务器控件：单个 RadioButton 控件或 RadioButtonList 控件。这两种控件都允许用户从一小组互相排斥的预定义选项中进行选择。这些控件允许定义任意数目带标签的单选按钮，并将它们水平或垂直排列。

每类控件都有各自的优点。单个 RadioButton 控件使用户可以更好地控制单选按钮组的布局。例如，可以在各单选按钮之间加入文本(即非单选按钮文本)。

RadioButtonList 控件不允许在按钮之间插入文本，但如果想将按钮绑定到数据源，使用这类控件要方便得多。在编写代码以检查所选定的按钮方面，它也稍微简单一些。

1. 对单选按钮分组

单选按钮很少单独使用，而是进行分组以提供一组互斥的选项。在一个组内，每次只能选择一个单选按钮。可以用下列方法创建分组的单选按钮：

(1) 先向页面中添加单个的 RadioButton 控件，然后将所有这些控件手动分配到一个组中。具有相同组名的所有单选按钮视为单个组的组成部分。

(2) 向页面中添加一个 RadioButtonList 控件。该控件中的列表项将自动进行分组。

2. RadioButton 事件

在单个 RadioButton 控件和 RadioButtonList 控件之间，事件的工作方式略有不同。

单个 RadioButton 控件在用户单击该控件时引发 CheckedChanged 事件。默认情况下，这一事件并不导致向服务器发送页面，但通过将 AutoPostBack 属性设置为 true，可以使该控件强制立即发送。

与单个的 RadioButton 控件相反，RadioButtonList 控件在用户更改列表中选定的单选按钮时会引发 SelectedIndexChanged 事件。默认情况下，此事件并不导致向服务器发送窗体，

但可以通过将 AutoPostBack 属性设置为 true 来指定此选项。

【例 4-8】演示单选按钮控件 RadioButton 和单选按钮列表控件 RadioButtonList 使用。

(1) 运行 VWD 2008，新建名为 RadioButtonListWebServerControl 的 ASP.NET 网站。

在网站的根目录下新建名为 shared 的文件夹，再在 shared 文件夹中新建名为 images 的文件夹，将准备好的图片复制到 images 文件夹中。

(2) 切换到 Default.aspx 页的【设计】视图，如图 4-12 所示。

(3) 输入静态文本“请输入您的信息：”，在【格式】工具栏中使用【块格式】下拉列表将该文本的格式设置为“标题 1”。

(4) 输入静态文本“姓名”、“性别”、“您想看的行星图片”和 RadioButtonList Example。

(5) 从工具栏中依次将控件拖动到设计视图，添加 TextBox1、RadioButton1 和 RadioButton2 这 3 个控件。

(6) 在 VWD 2008 中的【表】菜单上，单击【插入表】，出现【插入表格】对话框。

(7) 设置行数值为 1，列数值为 2，单元格衬距值为 2，单元格间距值为 0，边框粗细值为 1，边框颜色值为#FF00FF，单击【确定】按钮，【设计】视图上出现一个 1 行 2 列的表格。

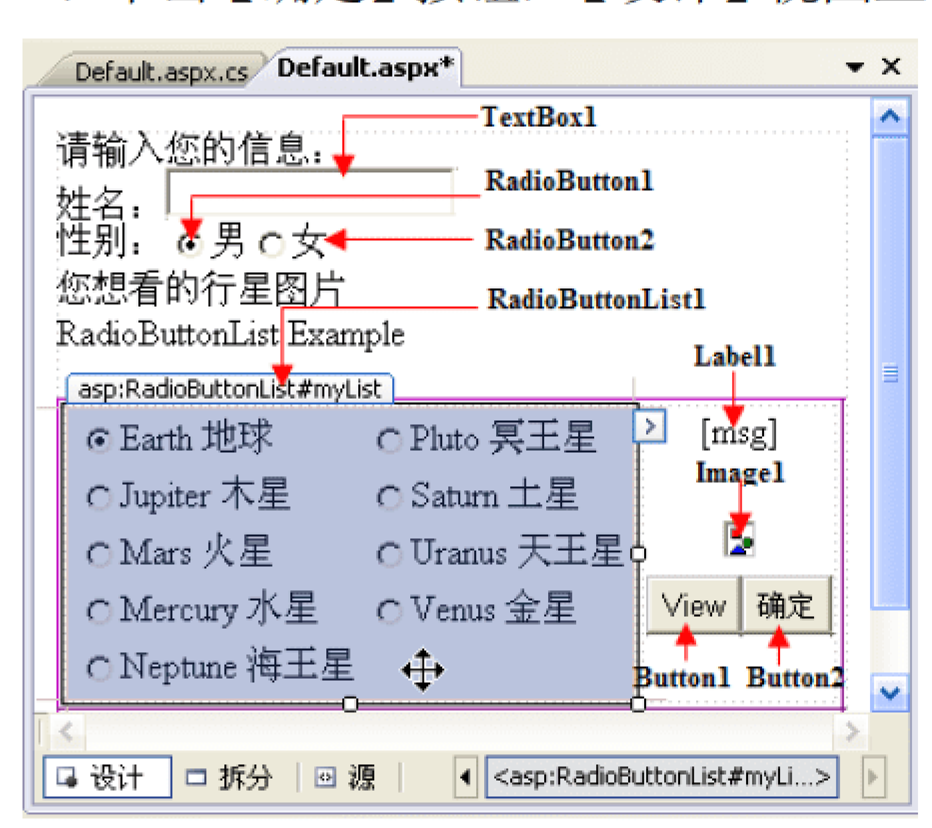


图 4-12　按钮控件

(8) 从工具栏中依次将控件拖动到表格的各单元格中，也可先在表格的各单元格中单击，再在工具栏中相应的控件上双击添加控件。在第 1 列中添加 RadioButtonList1 控件，在第 2 列中添加 Label1、Image1、Button1 和 Button2 控件。

(9) 属性设置。

RadioButton1 控件的属性设置为：ID="RadioButtonMale"、GroupName="sex"、Text="男"、Checked="True"；

RadioButton2 控件的属性设置为：ID="RadioButtonFemale"、GroupName="sex"、Text="女"；

RadioButtonList1 控件的属性设置为：id="myList"、RepeatColumns="2"；

Label1 控件的属性设置为：id="msg"；

Image1 控件的属性设置为：id="Image"；

Button1 控件的属性设置为：ID="Button1"、Text="View"；

Button2 控件的属性设置为：ID="ButtonOK"、onclick="ButtonOK_Click"、Text="确定"；

(10) 在【设计】视图中，单击 RadioButtonList1 控件右上角的【智能标记】，切换到

【RadioButtonList 任务】对话框，不选择【启用 AutoPostBack】，单击【编辑项...】，出现【ListItem 集合编辑器】，如图 4-13 所示，单击【添加】，添加成员 ListItem0～ListItem8，分别将其属性 Text 值设置为“Earth 地球”、“Jupiter 木星”、“Mars 火星”、“Mercury 水星”、“Neptune 海王星”、“Pluto 冥王星”、“Saturn 土星”、“Uranus 天王星”、“Venus 金星”，分别将其属性 Value 值设置为 earth.gif、jupiter.gif、mars.gif、mercury.gif、neptune.gif、pluto.gif、saturn.gif、uranus.gif、venus.gif。其主要的 HTML 代码如下所示。

```
<asp:RadioButtonList id="myList" runat="server" RepeatColumns="2">
    asp:listitem selected="True"   value="earth.gif" text="Earth 地球"/>
    <asp:listitem value="jupiter.gif" text="Jupiter 木星"/>
    <asp:listitem value="mars.gif" text="Mars 火星"/>
    <asp:listitem value="mercury.gif" text="Mercury 水星"/>
    <asp:listitem value="neptune.gif" text="Neptune 海王星"/>
    <asp:listitem value="pluto.gif" text="Pluto 冥王星"/>
    <asp:listitem value="saturn.gif" text="Saturn 土星"/>
    <asp:listitem value="uranus.gif" text="Uranus 天王星"/>
    <asp:listitem value="venus.gif" text="Venus 金星"/>
</asp:RadioButtonList>
```



图 4-13　为 RadioButtonList 控件添加成员

(11) 添加事件的处理程序代码。

```
Protected void ButtonOK_Click(object sender, EventArgs e)
{
    string str1, str2, str3= "";
    if (TextBox1.Text == "")
    {
        string   scriptString = "alert('用户名不能为空！'); ";
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(),"warning", scriptString, true);
    }
    else
    {
        str1 = TextBox1.Text;
        if (RadioButtonMale.Checked)
```

```
                { str2 = "男"; }
            else
                { str2 = "女"; }
            str3 = this .myList.SelectedItem.Text;
            string scriptContent = "alert('您提供的信息是：姓　名："+str1+
                "　性别："+str2+"  选择的行星图片是："+str3+" '); ";
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(),"success", scriptContent, true);
        }
    }
    void Page_Load ( object src, EventArgs e )
    {
    if (!Page.IsCallback)
    {
        if (myList.SelectedIndex > -1)
        {
            msg.Text = myList.SelectedItem.Text;
            Image.ImageUrl = "~/shared/images/" + myList.SelectedItem.Value;
            Image.AlternateText = myList.SelectedItem.Text;
        }
    }
```

(12) 按 Ctrl+F5，观看显示效果。

4.3.7　MultiView 和 View 控件

MultiView 和 View 控件可以制作出选项卡的效果，MultiView 控件用作一个或多个 View 控件的外部容器。View 控件又可包含标记和控件的任何组合。

如果要切换视图，可以使用控件的 ID 或者 View 控件的索引值。在 MultiView 控件中，一次只能将一个 View 控件定义为活动视图。如果某个 View 控件定义为活动视图，它所包含的子控件则会呈现到客户端。可以使用 ActiveViewIndex 属性或 SetActiveView 方法定义活动视图。如果 ActiveViewIndex 属性为空，则 MultiView 控件不向客户端呈现任何内容。如果活动视图设置为 MultiView 控件中不存在的 View，则会在运行时引发 ArgumentOutOfRangeException。下面介绍一些常用的属性、方法。

ActiveViewIndex 属性：用于获取或设置当前被激活显示的 View 控件的索引值。默认值为－1，表示没有 View 控件被激活。

SetActiveView 方法：用于激活显示特定的 View 控件。

ActiveViewChanged 事件：当视图切换时被激发。

MultiView 控件一次显示一个 View 控件，并公开该 View 控件内的标记和控件。通过设置 MultiView 控件的 ActiveViewIndex 属性，可以指定当前可见的 View 控件。

呈现 View 控件内容：未选择某个 View 控件时，该控件不会呈现到页面中。但是，每次呈现页面时都会创建所有 View 控件中的所有服务器控件的实例，并且将这些实例的值存储为页面的视图状态的一部分。

无论是 MultiView 控件还是各个 View 控件，除当前 View 控件的内容外，都不会在页面中显示任何标记。例如，这些控件不会以与 Panel 控件相同的方式来呈现 div 元素。这些控件也不支持可以作为一个整体应用于当前 View 控件的外观属性。但是可以将一个主题分配给 MultiView 或 View 控件，控件将该主题应用于当前 View 控件的所有子控件。

引用控件：每个 View 控件都支持 Controls 属性，该属性包含该 View 控件中的控件集合。也可以在代码中单独引用 View 控件中的控件。

在视图间导航：除了通过将 MultiView 控件的 ActiveViewIndex 属性设置为要显示的 View 控件的索引值可以在视图间导航外，MultiView 控件还支持可以添加到每个 View 控件的导航按钮。

若要创建导航按钮，可以向每个 View 控件添加一个按钮控件(Button、LinkButton 或 ImageButton)。然后可以将每个按钮的 CommandName 和 CommandArgument 属性设置为保留值以使 MultiView 控件移动到另一个视图。

【例 4-9】View 和 MultiView 控件示例。

(1) 运行 VWD 2008，新建名为 MultiViewWebServerControl 的 ASP.NET 网站。

(2) 在网站根目录下，添加新页面 MultiViewControl.aspx。

(3) 切换到 MultiViewControl.aspx 页的【设计】视图。

(4) 输入静态文本“按书名、类别或出版社搜索？”，如图 4-14 所示，添加 3 个 RadioButton 控件到页面上。切换到【源】视图，修改其 HTML 代码为：

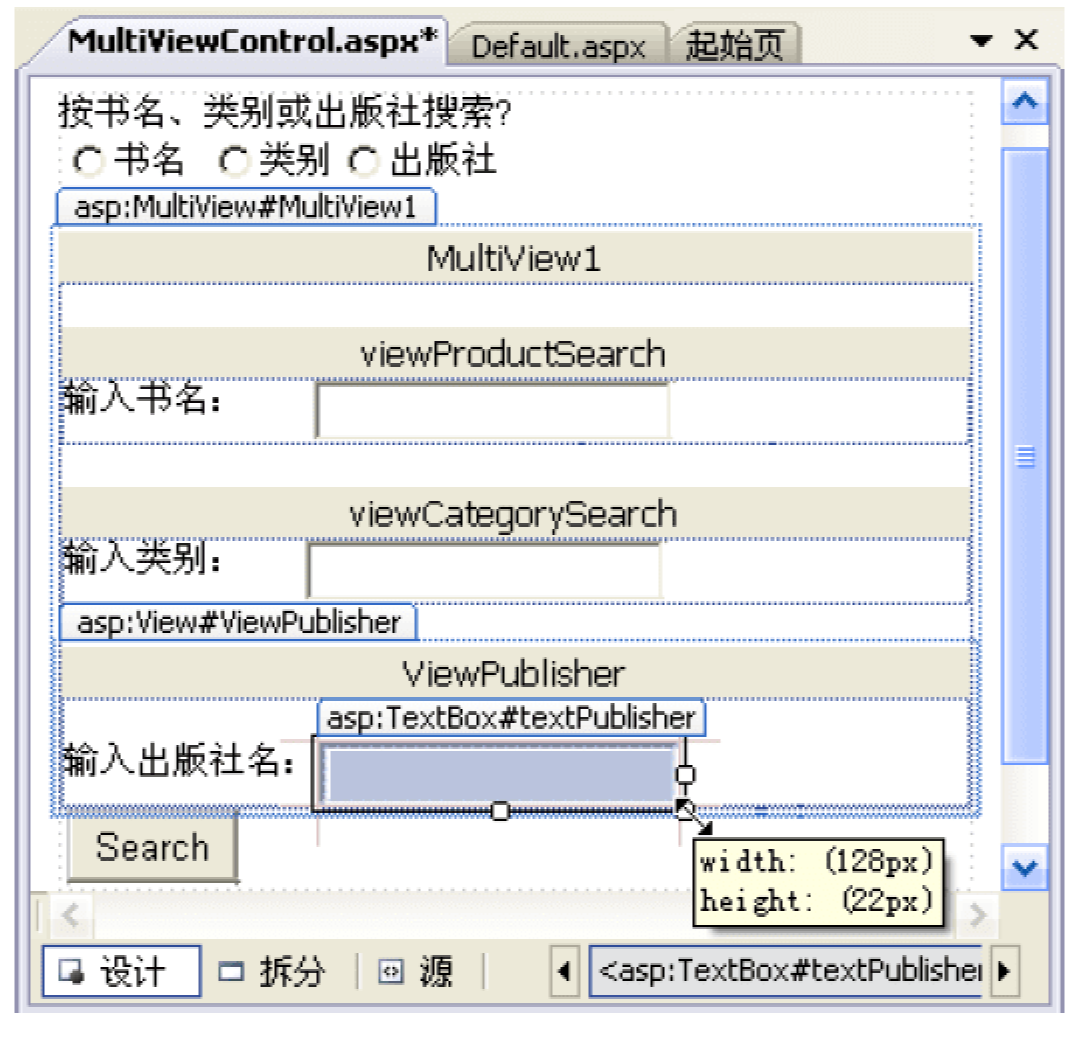


图 4-14　添加控件

```
<asp:RadioButton ID="radioProduct"    runat="server"    autopostback="true"
    GroupName="SearchType"    Text="书名"
    OnCheckedChanged="radioButton_CheckedChanged" />  
<asp:RadioButton ID="radioCategory"   runat="server"    autopostback="true"
    GroupName="SearchType"    Text="类别"
    OnCheckedChanged="radioButton_CheckedChanged" />
<asp:RadioButton ID="radioPublisher"   runat="server"   AutoPostBack="True"
```

```
GroupName="SearchType"    Text="出版社"
oncheckedchanged="radioButton_CheckedChanged" />
```

请注意将 3 个 RadioButton 的 CheckChanged 事件的处理程序设置为 oncheckedchanged="radioButton_CheckedChanged"，这样单击任一 RadioButton，响应它们的处理程序是相同的。

(5) 如图 4-14 所示，从工具箱的【标准】选项卡中，拖动【MultiView】控件到页面上，再拖到 3 个 View 控件到【MultiView】上，拖动一个 Button 控件到页面上。

分别单击三个 View 控件，将其 ID 属性分别改为 viewProductSearch、ViewCategorySearch、ViewPublisher；分别直接输入静态文本“输入书名”、“输入类别”、“输入出版社名”；从工具箱的“标准”选项卡中，拖动 3 个 Textbox 控件分布到 3 个 View 控件上，将其 ID 属性分别改为 textProductName、textCategory、textPublisher。

(6) 切换到【源】视图中，可看到如下所示的代码。

```
<asp:MultiView ID="MultiView1" runat="server"> <br />
    <asp:View ID="viewProductSearch" runat="server">
        输入书名：  <asp:TextBox ID="textProductName" runat="server"> </asp:TextBox>
    </asp:View> <br />
    <asp:View ID="viewCategorySearch" runat="server">
        输入类别：<asp:TextBox ID="textCategory" runat="server"> </asp:TextBox>
    </asp:View> <br />
    <asp:View ID="ViewPublisher" runat="server">
        输入出版社名：<asp:TextBox ID="textPublisher" runat="server"></asp:TextBox>
    </asp:View>
</asp:MultiView>
```

(7) 设置 Button1 控件的标记如下所示：

```
<asp:Button ID="btnSearch" OnClick="Button1_Click" runat="server" Text="Search" />
```

(8) 切换到 MultiViewControl.aspx.cs，在“类”体内添加如下所示的代码。

```
public enum SearchType
{
    NotSet = -1,
    Products = 0,
    Category = 1,
    Publisher = 2
}
protected void Page_Load(object sender, EventArgs e)
{
    radioProduct.Checked = true;
    MultiView1.ActiveViewIndex = 0;
}
protected void Button1_Click(Object sender, System.EventArgs e)
```

```
{
    if (MultiView1.ActiveViewIndex > -1)
    {
        SearchType mSearchType = (SearchType)MultiView1.ActiveViewIndex;
        switch (mSearchType)
        {
            case SearchType.Products:
                DoSearch(textProductName.Text, mSearchType);
                break;
            case SearchType.Category:
                DoSearch(textCategory.Text, mSearchType);
                break;
            case SearchType.Publisher:
                DoSearch(textPublisher.Text, mSearchType);
                break;
            case SearchType.NotSet:
                break;
       }
    }
}

protected void DoSearch(String searchTerm, SearchType type)
{
    // Code here to perform a search.
    string scriptString = "alert('"+"您输入的"+searchTerm+"');";
    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", scriptString, true);
    // Response.Write("您输入的"+ searchTerm );
}

protected void radioButton_CheckedChanged(Object sender, System.EventArgs e)
{
    if (radioProduct.Checked)
    {
        MultiView1.ActiveViewIndex = (int)SearchType.Products;
    }
    else if (radioCategory.Checked)
    {
        MultiView1.ActiveViewIndex = (int)SearchType.Category;
    }
    else if (radioPublisher.Checked)
    {
        MultiView1.ActiveViewIndex = (int)SearchType.Publisher;
    }
}
```

(9) 按 Ctrl+F5，观看程序运行效果。

4.3.8　AdRotator(广告控件)服务器控件

AdRotator 服务器控件提供一种在 ASP.NET 网页上显示广告的方法。该控件可显示.gif 文件或其他图形图像。当用户单击广告时，系统会将他们重定向到指定的目标 URL。

AdRotator 服务器控件可从数据源(通常是 XML 文件或数据库表)提供的广告列表中自动读取广告信息，如图形文件名和目标 URL。用户可以将信息存储在一个 XML 文件或数据库表中，然后将 AdRotator 控件绑定到该文件。

AdRotator 控件会随机选择广告，每次刷新页面时都将更改显示的广告。广告可以加权以控制广告条的优先级别，这可以使某些广告的显示频率比其他广告高。也能编写在广告间循环的自定义逻辑。

AdRotator 控件的所有属性都是可选的。XML 文件中可以包括下列属性。

- ImageUrl：要显示的图像的 URL。
- NavigateUrl：单击 AdRotator 控件时要转到的网页的 URL。
- AlternateText：图像不可用时显示的文本。
- Keyword：可用于筛选特定广告的广告类别。
- Impressions：一个指示广告的可能显示频率的数值(加权数值)。在 XML 文件中，所有 Impressions 值的总和不能超过 2,048,000,000 - 1。
- Height：广告的高度(以像素为单位)。此值会重写 AdRotator 控件的默认高度设置。
- Width：广告的宽度(以像素为单位)。此值会重写 AdRotator 控件的默认宽度设置。

【例 4-10】使用 AdRotator 服务器控件显示数据库中的广告。

(1) 新建名为 AdRotatorXmlSource 文件系统网站。

(2) 在网站根目录下新建名为 Images 的文件夹，将准备好的广告图片复制到该文件夹。

(3) 在网站的 App_Data 文件夹中新建名为 AdDataSource.ads 的 XML 文件。

为了更好地保证安全，请将该文件的文件扩展名命名为除.xml 之外的名称，如.ads。建议将广告文件放置在 App_Data 文件夹中，因为 ASP.NET 可防止浏览器利用该文件夹中的文件。

(4) 向广告信息文件中添加下列 XML 元素：

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements xmlns="http://schemas.microsoft.com/AspNet/AdRotator-Schedule-File">
</Advertisements>
```

(5) 在 Advertisements 元素中，为每个要包括在广告列表中的广告创建一个 Ad 元素，然后保存文件。

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements xmlns="http://schemas.microsoft.com/AspNet/AdRotator-Schedule-File">
  <Ad>
    <ImageUrl>~/images/baidu.gif</ImageUrl>
```

```
        <NavigateUrl>http://www.baidu.com</NavigateUrl>
        <AlternateText>Ad for baidu, Ltd. Web site</AlternateText>
        <Impressions>100</Impressions>
    </Ad>
    <Ad>
        <ImageUrl>~/images/google.gif</ImageUrl>
        <NavigateUrl>http://www.google.com</NavigateUrl>
        <AlternateText>Ad for Google Web site</AlternateText>
        <Impressions>50</Impressions>
    </Ad>
</Advertisements>
```

(6) 切换到 default.aspx，在网页上要显示广告的位置放置一个 AdRotator。如图 4-15 所示。



图 4-15　AdRotator 控件

(7) 单击 AdRotator 控件的【智能标记】，选择【新建数据源…】，选择【xml 文件】，单击【确定】，在【配置数据源】对话框中，将【数据文件】输入框设置为~/App_Data/AdDataSource.ads，最后单击【确定】。

(8) 测试广告。按 Ctrl+F5 运行该页。单击几次浏览器的【刷新】按钮可显示不同的广告信息。出现的广告是随机变化的。

4.3.9　Literal 控件和 Panel 控件

Literal 控件和 Panel 控件均可作为容器控件，但二者的适用场合不同，下面分别介绍。

1. Literal 控件

Literal 控件可以作为页面上其他内容的容器，最常用于向页面中动态添加内容。

对于静态内容，无需使用容器，可以将标记作为 HTML 直接添加到页面中。但是，如果要动态添加内容，则必须将内容添加到容器中。典型的容器有 Label 控件、Literal 控件、Panel 控件和 PlaceHolder 控件。

Literal 控件与 Label 控件的区别在于 Literal 控件不向文本中添加任何 HTML 元素。(Label 控件呈现一个 span 元素。)因此，Literal 控件不支持包括位置属性在内的任何样式属性。但是，Literal 控件允许指定是否对内容进行编码。

Panel 和 PlaceHolder 控件呈现为 div 元素，这将在页面中创建离散块，与 Label 和 Literal

控件进行内嵌呈现的方式不同。

通常情况下，当希望文本和控件直接呈现在页面中而不使用任何附加标记时，可使用 Literal 控件。

Literal 控件常用的属性是 Mode 属性，该属性用于指定控件对用户所添加的标记的处理方式。可以将 Mode 属性设置为以下值。

- Transform.：将对添加到控件中的任何标记进行转换，以适应请求浏览器的协议。如果向使用 HTML 外的其他协议的移动设备呈现内容，此设置非常有用。
- PassThrough.：添加到控件中的任何标记都将按原样呈现在浏览器中。
- Encode：将使用 HtmlEncode 方法对添加到控件中的任何标记进行编码，这会将 HTML 编码转换为其文本表示形式。例如，<b>标记将呈现为<b>。当希望浏览器显示而不解释标记时，编码将很有用。编码对于安全也很有用，有助于防止在浏览器中执行恶意标记。显示来自不受信任的源的字符串时推荐使用此设置。

【例 4-11】演示 Literal 控件的使用。

(1) 运行 VWD 2008，新建一个名为 LiteralControl 的 ASP.NET 网站。

(2) 向 default.aspx 页面添加 2 个 RadioButton 控件，1 个 Literal 控件，1 个 Label 控件。按如下所示的标记代码设置 2 个 RadioButton 控件的相关属性。

```
<asp:RadioButton ID="radioEncode" runat="server" GroupName="LiteralMode" Checked="True"
    Text="Encode" AutoPostBack="True" />
<asp:RadioButton ID="radioPassthrough" runat="server" GroupName="LiteralMode"
    Text="PassThrough" AutoPostBack="True" />
```

(3) 添加 Page_Load 事件代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    Literal1.Text = "This <b>text</b> is inserted dynamically.";
    Label1 .Text ="This <b>text</b> is inserted dynamically.";
    if (radioEncode.Checked == true)
    {
        Literal1.Mode = LiteralMode.Encode;
    }
    if (radioPassthrough.Checked == true)
    {
        Literal1.Mode = LiteralMode.PassThrough;
    }
}
```

(4) 按 Ctrl+F5 运行该页，效果如图 4-16 所示。选择不同的单选按钮，观察文本的变化。

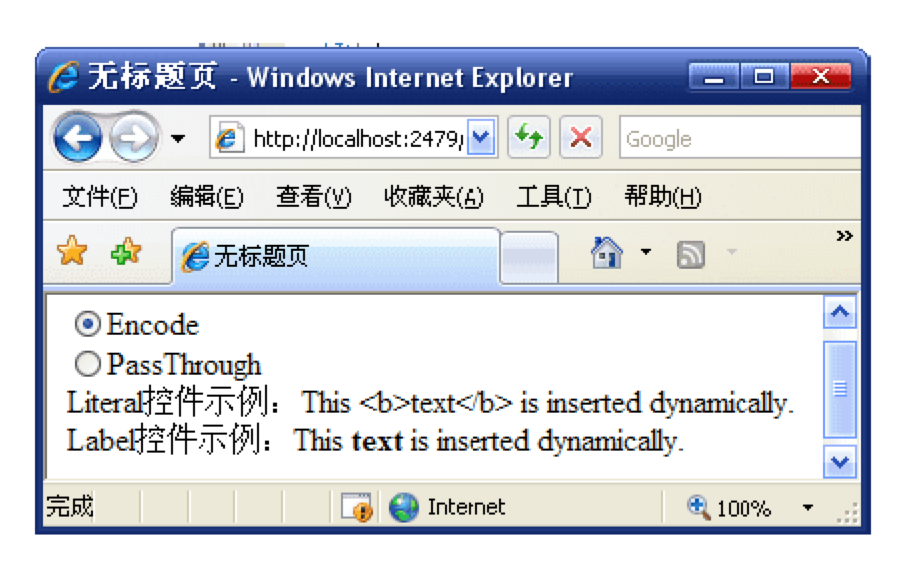


图 4-16　Literal 控件的运行示例

2. Panel 控件

Panel 控件在 ASP.NET 网页内提供了一种容器控件，可以将它用作静态文本和其他控件的父控件，向该控件添加其他控件和静态文本。

可以将 Panel 控件用作其他控件的容器。当以编程方法创建内容并需要一种将内容插入到页面中的方法时，此方法尤为适用。以下部分描述了可以使用 Panel 控件的其他方法。

(1) 动态生成的控件的容器

Panel 控件为在运行时创建的控件提供了一个方便的容器。

(2) 对控件和标记进行分组

对于一组控件和相关的标记，可以通过把其放置在 Panel 控件中，然后操作此 Panel 控件的方式将它们作为一个单元进行管理。例如，可以通过设置面板的 Visible 属性来隐藏或显示该面板中的一组控件。

(3) 具有默认按钮的窗体

可将 TextBox 控件和 Button 控件放置在 Panel 控件中，然后通过将 Panel 控件的 DefaultButton 属性设置为面板中某个按钮的 ID 来定义一个默认的按钮。如果用户在面板内的文本框中进行输入时按 Enter，这与用户单击特定的默认按钮具有相同的效果。这有助于用户更有效地使用项目窗体。

(4) 向其他控件添加滚动条

有些控件(如 TreeView 控件)没有内置的滚动条。通过在 Panel 控件中放置滚动条控件，可以添加滚动行为。若要向 Panel 控件添加滚动条，请设置 Height 和 Width 属性，将 Panel 控件限制为特定的大小，然后再设置 ScrollBars 属性。

(5) 页上的自定义区域

可使用 Panel 控件在页面上创建具有自定义外观和行为的区域，如下所示。

- 创建一个带标题的分组框：可设置 GroupingText 属性来显示标题。呈现页时，Panel 控件的周围将显示一个包含标题的框，其标题是用户指定的文本。

说明：不能在 Panel 控件中同时指定滚动条和分组文本。如果设置了分组文本，其优先级高于滚动条。

- 在页面上创建具有自定义颜色或其他外观的区域：Panel 控件支持外观属性(如 BackColor 和 BorderWidth)，可以设置外观属性为页面上的某个区域创建独特的外观。

说明：设置 GroupingText 属性将自动在 Panel 控件周围呈现一个边框。

【例 4-12】演示文件 Panel 控件的使用。

(1) 运行 VWD 2008，新建名为 PanelWebServerControl 的 ASP.NET 网站。

(2) 新建页面 PanelExample.aspx，切换到其【设计】视图。

(3) 从工具箱的【标准】选项卡中，将 Panel 控件拖到页面上，如图 4-17 所示。

(4) 添加 DropDownList1、CheckBox1、CheckBox2、Button1、Literal1、Literal2 控件，如图 4-17 所示。

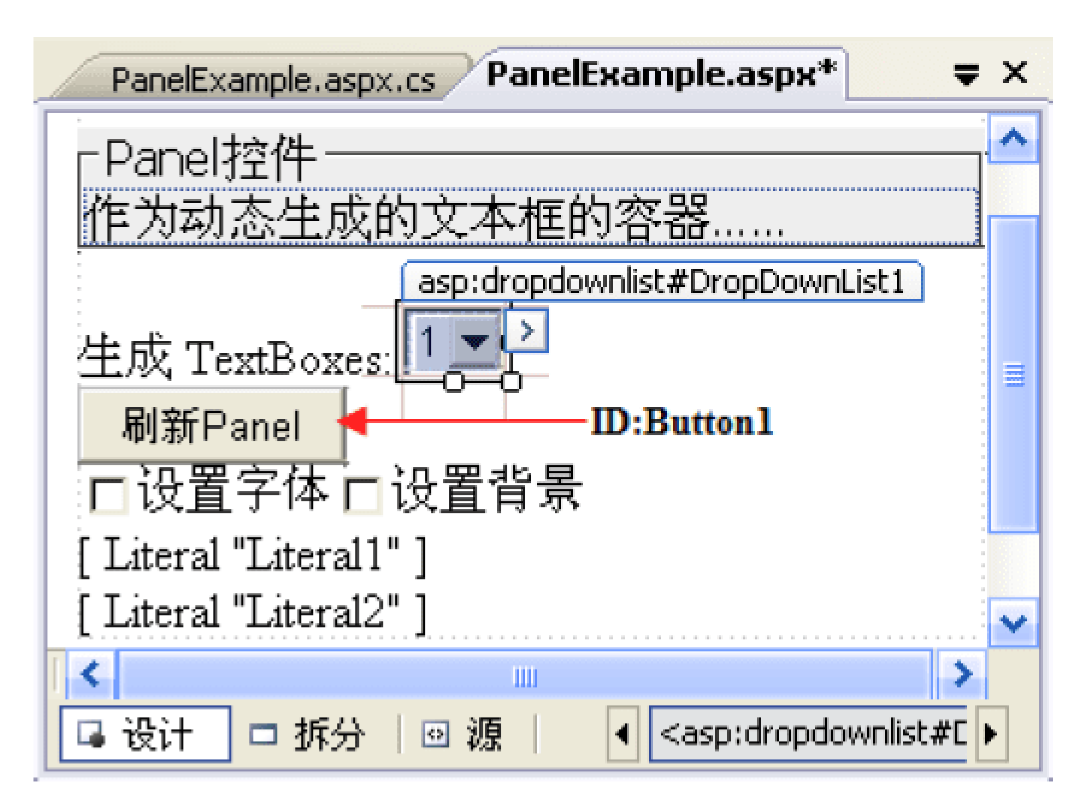


图 4-17　添加 Panel 等控件

(5) 切换到PanelExample.aspx页的【源】视图，根据下面的代码设置各个控件的属性。

```
<asp:panel id="myPanel" runat="server" backcolor="#eeeeee" width="300px"
    GroupingText="Panel 控件">
    <p>作为动态生成的文本框的容器······ </p>
</asp:panel>
生成 TextBoxes:
<asp:dropdownlist id="DropDownList1" runat="server">
   <asp:listitem value="1">1</asp:listitem>
   <asp:listitem value="2">2</asp:listitem>
   <asp:listitem value="3">3</asp:listitem>
</asp:dropdownlist>
<asp:button ID="Button1" text="刷新 Panel" runat="server" onclick="Button1_Click" />
<asp:CheckBox ID="CheckBoxChangeFont" runat="server" AutoPostBack="True"
    oncheckedchanged="CheckBoxChangeFont_CheckedChanged" Text="设置字体"/>
<asp:CheckBox ID="CheckBoxChangeBkGround" runat="server" AutoPostBack="True"
   oncheckedchanged="CheckBoxChangeBkGround_CheckedChanged" Text="设置背景"/>
```

(6) 切换到 PanelExample.aspx.cs 页的【代码】视图，根据下面的代码设置各控件的 Click 事件的处理程序。

```
void Page_Load(object src, EventArgs e)
{
    // generate textbox controls
    int numtexts = int.Parse(DropDownList1.SelectedItem.Value);
    for (int i = 1; i <= numtexts; i++)
```

```
        {
            myPanel.Controls.Add(new LiteralControl("<p>"));
            TextBox t = new TextBox();
            t.Text = "TextBox" + i.ToString();
            t.ID = "TextBox" + i.ToString();
            myPanel.Controls.Add(t);
        }
    }
    protected void CheckBoxChangeFont_CheckedChanged(object sender, EventArgs e)
    {
        if (CheckBoxChangeFont.Checked)
        {
            this.myPanel.Font.Italic = true;
            this.myPanel.ForeColor = System.Drawing.Color.Red ;
            Literal1 .Text ="当前所显示字型是“斜体”，颜色是“红色”";
        }
        else
        {
            this.myPanel.Font.Italic = false ;
            this.myPanel.ForeColor = System.Drawing.Color.Blue;
            Literal1.Text = "当前所显示字型是“默认字体”，颜色是“蓝色”";
        }
    }
    protected void CheckBoxChangeBkGround_CheckedChanged(object sender, EventArgs e)
    {
        if (CheckBoxChangeBkGround.Checked)
        {
            this.myPanel.BackColor   = System.Drawing.Color.Bisque;//Bisque 橘黄色
            Literal2.Text = "当前所显示背景颜色是“Bisque 橘黄色”。";
        }
        else
        {
            this.myPanel.BackColor = System.Drawing.Color.Beige;//Beige  米黄色
            Literal2.Text = "当前所显示背景颜色是“Beige  米黄色”。";
        }
    }
```

(7) 按 Ctrl+F5，观看显示效果。

4.4　验证控件

ASP.NET 为开发人员提供了一套完整的服务器控件来验证用户输入的信息是否有效，这些控件可与 ASP.NET 网页上的任何控件(包括 HTML 和服务器控件)一起使用。

4.4.1　ASP.NET 验证控件

表 4-2 列出了 ASP.NET 验证控件及其功能说明。

表 4-2　ASP.NET 验证控件

验证类型	使用的控件	说　明
必选项	必选项验证控件 RequiredFieldValidator	验证一个必填字段，如果这个字段没填，那么，将不能提交信息
与某值的比较	比较验证控件 CompareValidator	比较验证：将用户输入与一个常数值或者另一个控件或特定数据类型的值进行比较(使用小于、等于或大于等比较运算符)，同时也可以用来校验控件中内容的数据类型：如整形、字符串型等。如密码和确认密码两个字段是否相等
范围检查	范围验证控件 RangeValidator	范围验证：RangeValidator 控件可以用来判断用户输入的值是否在某一特定范围内。可以检查数字对、字母对和日期对限定的范围。属性 MaximumValue 和 MinimumValue 用来设定范围的最大和最小值
模式匹配	正则表达式验证控件 RegularExpressionValidator	正则表达式验证：它根据正则表达式来验证用户输入字段的格式是否合法，如电子邮件、身份证、电话号码等。ControlToValidate 属性选择需要验证的控件，ValidationExpression 属性则编写需要验证的表达式的样式
用户定义	用户定义验证控件 CustomValidator	使用自己编写的验证逻辑检查用户输入。此类验证能够检查在运行时派生的值。在运行定制的客户端 JavaScript 或 VBScript 函数时，可以使用这个控件
验证汇总	验证汇总控件 ValidationSummary	验证汇总：该控件不执行验证，但该控件将本页所有验证控件的验证错误信息汇总为一个列表并集中显示，列表的显示方式由 DisplayMode 属性设置

- 验证对象模型：通过使用由各个验证控件和页面公开的对象模型，可以与验证控件进行交互。每个验证控件都会公开自己的 IsValid 属性，可以测试该属性以确定该控件是否通过验证测试。页面还公开一个 IsValid 属性，该属性总结页面上所有验证控件的 IsValid 状态。该属性允许您执行单个测试，以确定是否可以继续自行处理。页面还公开一个包含页面上所有验证控件的列表的 Validators 集合。可以依次通过这一集合来检查单个验证控件的状态。
- Button 控件和验证：如果某一页面包含 ASP.NET 验证控件，则在默认情况下，单击按钮控件会导致验证控件执行检查。如果为验证控件启用了客户端验证，则在验证检查失败时不会提交该页面。表 4-3 描述了按钮控件所支持的可以更精确地控制验证过程的属性。

表 4-3　Button 控件的部分属性

属　性	说　明
CausesValidation	指定单击按钮是否还执行验证检查。将此属性设置为 false 可避免验证检查
ValidationGroup	能够指定单击按钮时调用页面上的哪些验证程序。如果未建立任何验证组，则单击按钮会调用页面上的所有验证程序

4.4.2　验证 Web 窗体页中的用户输入

本示例演示如何使用 ASP.NET 验证控件检查网页中的用户输入。

【例 4-13】验证 Web 窗体页中的用户输入，【设计】视图如图 4-18 所示。



图 4-18　添加验证控件

(1) 打开 VWD 2008，新建网站 ValidationWebServerControl。

(2) 切换到 Default.aspx【设计】视图，输入页标题“提交预订”。

添加 3 个 TextBox 控件，ID 属性分别改为 textEmail、textNumberInParty 和 textPreferredDate；添加 Button 控件，属性设置为 ID="buttonSubmit"、Text="提交申请"、ValidationGroup="AllValidators"；添加 Label 控件，属性设置为 ID="labelMessage"、Text=""。

(3) 如图 4-18 所示，在上述文本框左侧输入文本作为标题。

(4) 双击【提交申请】按钮，为其 Click 事件创建一个处理程序，添加以下代码：

```
protected void buttonSubmit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        labelMessage.Text = "您的预订已处理！ ";
        labelMessage.ForeColor = System.Drawing.Color.Red;
    }
}
```

该按钮处理程序只显示一条消息，它不执行实际的处理操作。但显示该消息将允许在本示例的后面部分测试验证结果。

(5) 切换到【设计】视图，双击页面空白区域以创建一个 Page_Load 处理程序，添加以下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    labelMessage.Text = "";
}
```

在通过所有验证后，该代码将清除该页上所显示的消息。该代码对本示例后面部分的测试很有用。

(6) 添加 RequiredFieldValidator 并将该控件放在 textEmail 文本框右侧，设置 RequiredFieldValidator 控件的以下属性，如表 4-4 所示。

表 4-4　RequiredFieldValidator 控件的属性设置

属　性	值	说　明
ControlToValidate	textEmail	将该验证控件绑定到要验证其内容的文本框
Display	Dynamic	指定只在显示错误需要时才呈现该控件
ErrorMessage	须填电子邮件地址	显示摘要错误中的文本
Text	*	星号表示某字段是必填的。只在出现错误时，才显示该文本
ValidationGroup	AllValidators	将验证控件分组

(7) 添加 RegularExpressionValidator 控件并将其放在刚添加的 RequiredFieldValidator 右侧，设置 RegularExpressionValidator 控件属性为：

ControlToValidate="textEmail"、Display="Dynamic"、ErrorMessage="电子邮件地址必须采用 name@domain.xyz 格式"、Text="无效格式！"、ValidationGroup="AllValidators"。

(8) 在 RegularExpressionValidator 正则表达式验证控件仍处于选定状态时，在【属性】窗口中单击 ValidationExpression 框中的省略号按钮，在【标准表达式】列表中，单击【Internet 电子邮件地址】，单击【确定】关闭对话框。

(9) 添加 RequiredFieldValidator 控件，将该控件绑定到 textNumberInParty 文本框并将其 ErrorMessage 属性设置为“请指出团队人数”。

(10) 添加 RangeValidator 控件并将其放在刚添加的 RequiredFieldValidator 控件右侧，设置 RangeValidator 控件的以下属性：

ControlToValidate="textNumberInParty" ， Display="Dynamic" ，MaximumValue="20"，MinimumValue="1"，ErrorMessage="为团队人数输入一个介于 1 和 20 之间的数字"，ValidationGroup="AllValidators"，Type="Integer"，Text="输入一个介于 1 和 20 之间的数字"

RangeValidator 控件执行下面两个功能：首先确保用户所输入的数据是数字，然后检查该数字是否在指定的最小值和最大值之间。

(11) 按 Ctrl+F5 运行该页。分别输入正确的、不正确的电子邮件地址，在指定范围内和

不在指定范围内的数字，进行验证。

(12) 从工具箱的【验证】组中，将 ValidationSummary 控件拖动到页面上，将其 ValidationGroup 属性设置为 AllValidators。运行该页，执行相同的测试。

对于每条错误，将看到在两个位置中显示了错误信息。在验证控件所在的位置上显示了一条简短的错误信息(验证程序的 Text 属性值)，在 ValidationSummary 控件中显示了一条较长的错误信息(该控件的 ErrorMessage 属性值)。

(13) 将 ValidationSummary 控件的 ShowMessageBox 属性设置为 true。运行该页。执行相同的测试。此时，每条错误将会导致在浏览器中显示一个弹出消息框。

(14) 添加 CustomValidator(自定义验证控件)放在 textPreferredDate 文本框右侧，并设置 CustomValidator 控件属性为：

ID="CustomValidator1"， ControlToValidate="textPreferredDate"， Display="Dynamic"，
ErrorMessage="以 yyyy-m-d 格式输入一个日期"， Tex="无效日期格式(需要 yyyy-m-d)"
EnableClientScript="False"，ValidationGroup="AllValidators"(禁用客户端验证)。

(15) 双击 CustomValidator 控件，为其 ServerValidate 事件创建一个处理程序，添加以下代码：

```
protected void CustomValidator1_ServerValidate(object source,ServerValidateEventArgs args)
{
  try
  {
     DateTimeFormatInfo info1 = (DateTimeFormatInfo)
         Thread.CurrentThread.CurrentCulture.GetFormat(typeof(DateTimeFormatInfo));
     DateTime time1 = DateTime.ParseExact(args.Value, "d", info1,
         DateTimeStyles.AllowWhiteSpaces);
     args.IsValid = true;
  }
   catch
   {
      args.IsValid = false;
    }
}
```

在用户提交此页面时运行该代码。该代码使用传入处理程序的 ServerValidateEventArgs (args)对象与该验证控件交互。用户在 textPreferredDate 文本框中所输入的值作为 args 对象的 Value 属性传递。在检查用户输入是否有效后，将 args 对象的 IsValid 属性相应地设置为 true 或 false。如果将该属性设置为 false，则该验证程序将显示其错误信息。

(16) 在 Button_Click 处理程序中，添加以下代码：

```
protected void buttonSubmit_Click(object sender, EventArgs e)
{
    if(Page.IsValid)
```

```
        {
            labelMessage.Text = "您的预订已受理。";
            //Your reservation has been processed.
        }
        else
            labelMessage.Text = "页面不合语法。";//Page is not valid.
        }
    }
```

在使用时，必须在服务器端的任何处理期间都检查 Page.IsValid 属性以确定是否通过了所有验证检查。IsValid 属性返回该页面上所有验证控件的累积状态。此属性用于确保任何服务器端的处理逻辑都已通过所有验证检查。

(17) 按 Ctrl+F5 运行该页，测试 CustomValidator 控件。

【例 4-14】演示 CompareValidator 比较验证控件的使用方法。

(1) 新建名字为 CompareValidatorControl 的网站。

(2) 在设计【视图】中，添加如图 4-19 所示的控件，2 个 TextBox 控件，2 个 RequiredFieldValidator 控件，1 个 CompareValidator 控件。

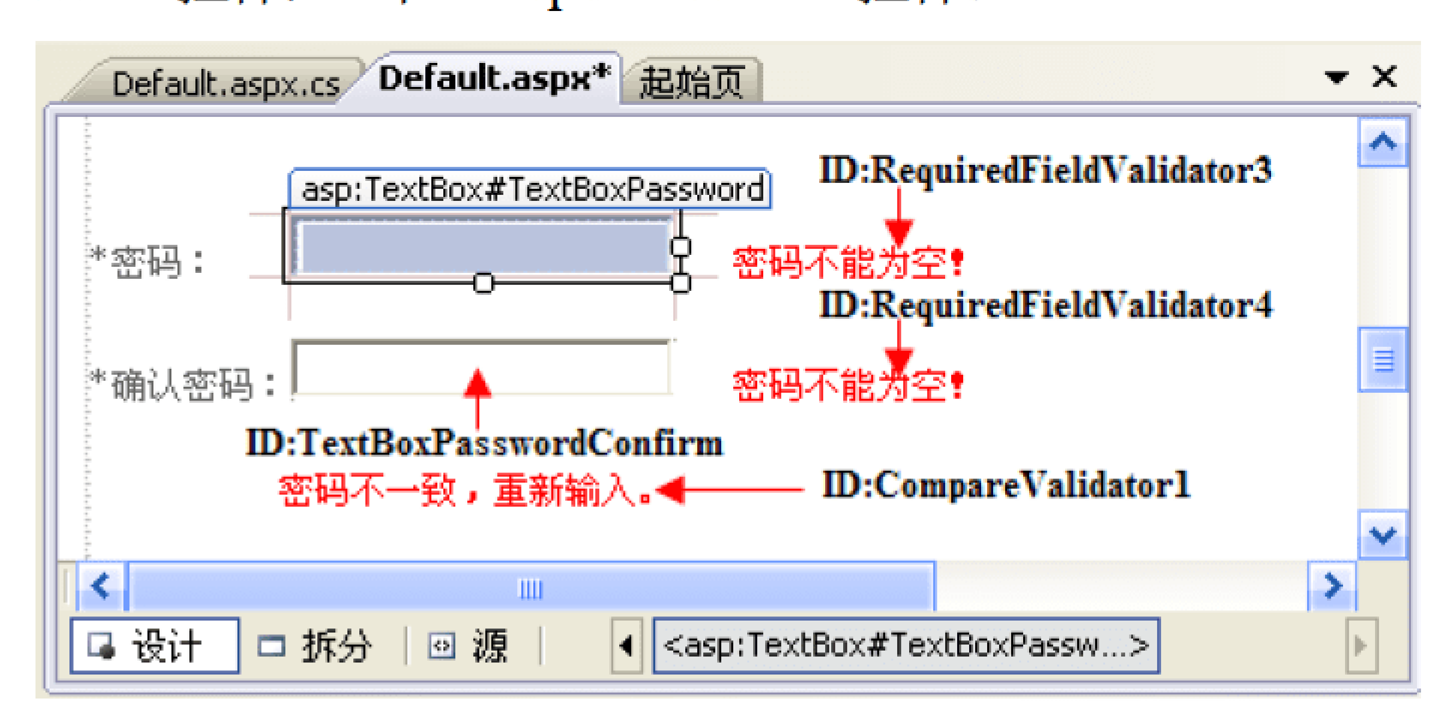


图 4-19　CompareValidator 控件

(3) 在 default.aspx 应该出现如下代码。

```
*密码:
<asp:TextBox ID="TextBoxPassword"    runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
    ControlToValidate="TextBoxPassword" ErrorMessage="密码不能为空！"
    Display="Dynamic">密码不能为空!
</asp:RequiredFieldValidator> <br />
*确认密码:
<asp:TextBox ID="TextBoxPasswordConfirm" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
    ControlToValidate="TextBoxPasswordConfirm" ErrorMessage="密码不能为空！"
    Display="Dynamic">密码不能为空!
</asp:RequiredFieldValidator> <br /><br />
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToCompare="TextBoxPassword"    ControlToValidate="TextBoxPasswordConfirm"
```

```
    ErrorMessage="密码不一致，重新输入。" Display="Dynamic">密码不一致，重新输入。
</asp:CompareValidator>
```

(4) 按 Ctrl+F5 运行该页，测试 CompareValidator 控件。

4.5　用户控件

有时可能需要控件具有 ASP.NET 内置服务器控件没有的功能。在这种情况下，用户可以创建自己的控件。有两个选择，可以创建用户控件和自定义控件。

用户控件是能够在其中放置标记和服务器控件的容器。然后，可以将用户控件作为一个单元对待，为其定义属性和方法。

自定义控件是编写的一个类，此类从 Control 或 WebControl 派生。

创建用户控件要比创建自定义控件方便很多，因为可以重用现有的控件。用户控件使创建具有复杂用户界面元素的控件极为方便。

1. 用户控件结构

ASP.NET Web 用户控件与完整的 ASP.NET 网页(.aspx 文件)相似，同时具有用户界面页和代码页。可以采取与创建 ASP.NET 页相似的方式创建用户控件，然后向其中添加所需的标记和子控件。用户控件可以像页面一样包含对其内容进行操作(包括执行数据绑定等任务)的代码。

2. 用户控件与 ASP.NET 网页有以下区别

(1) 用户控件的文件扩展名为.ascx。

(2) 用户控件中没有@Page 指令，而是包含@Control 指令，该指令对配置及其他属性进行定义。

(3) 用户控件不能作为独立文件运行。而必须像处理任何控件一样，将它们添加到 ASP.NET 页中。

(4) 用户控件中没有 HTML、body 或 form 元素。这些元素必须位于宿主页中。

(5) 可以在用户控件上使用与在 ASP.NET 网页上所用相同的 HTML 元素(HTML、body 或 form 元素除外)和 Web 控件。例如，如果要创建一个将用作工具栏的用户控件，则可以将一系列 Button 服务器控件放在该控件上，并创建这些按钮的事件处理程序。

【例 4-15】演示一个实现微调控件的用户控件。在此微调控件中，用户可以单击“向上”和“向下”按钮以滚动文本框中的一系列选择。

(1) 运行 VWD 2008，新建名为 WebUserControl 的 ASP.NET 网站。

(2) 在【解决方案资源管理器】中，用鼠标右键单击网站名 WebUserControl，选择【添加新项...】命令，在弹出的对话框中选择【Web 用户控件】模板，使用默认名称为 WebUserControl1.ascx，【语言】选择 Visual C#，选中【将代码放在单独的文件中】复选框，单击【添加】按钮，用户控件文件 WebUserControl.ascx 就添加到解决方案中了。

(3) 单击 WebUserControl1 的【设计】标签，切换到如图 4-20 所示的【设计】视图，然后在【工具箱】中依次双击【TextBox】控件、【Button】控件，添加 1 个 TextBox 控件，2 个 Button 控件。TextBox 控件的 ReadOnly 属性设置为 True，ID 属性设置为 TextBoxColor；其中一个 Button 控件的 ID 属性设置为 ButtonUp，Text 属性设置为 Up；另一个 Button 控件的 ID 属性设置为 ButtonDown，Text 属性设置为 Down。

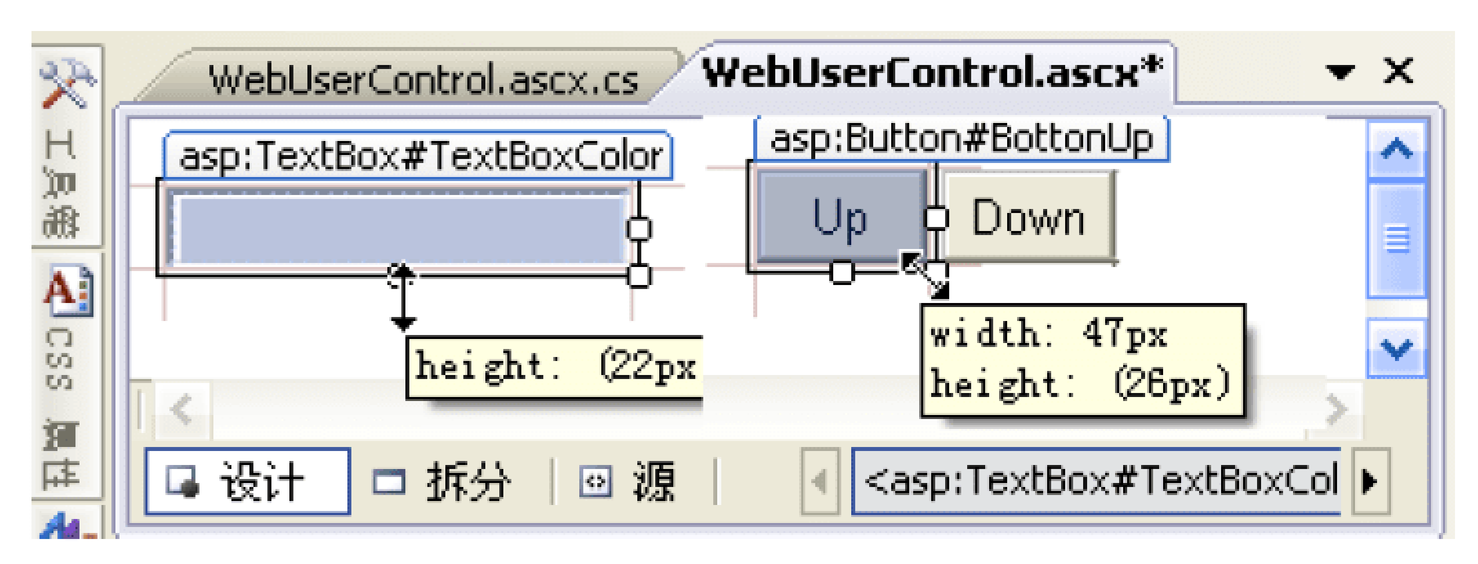


图 4-20　创建数据库文件夹

(4) 在 WebUserControl 的【设计】视图中双击，切换到 WebUserControl1 的代码视图，即打开 WebUserControl.ascx.cs 文件，定义变量，为 Page_Load 事件、Up 按钮和 Down 按钮的 Click 事件添加如下代码。

```
public partial class WebUserControl : System.Web.UI.UserControl
{
    protected int currentColorIndex;
    protected string[] colors ={ "Red", "Green", "Blue", "Yellow", };
    protected void Page_Load(object sender, EventArgs e)
    {//IsPostBack 的值指示是正为响应客户端回发而加载用户控件，还是正第一次加载和访问、
     //用户控件。如果是正为响应客户端回发而加载用户控件，则为 true；否则为 false。
        if (IsPostBack)
        {
            currentColorIndex = Int16.Parse(ViewState["currentColorIndex"].ToString());
        }
        else
        {
            currentColorIndex = 0;
            DisplayColor();
        }
    }
    private void DisplayColor()
    {
        TextBoxColor.Text = colors[currentColorIndex];
      //在文本框文字改变时字体的颜色也相应改变
      TextBoxColor.ForeColor = System.Drawing.Color.FromName(colors[currentColorIndex]);
      String strColor = colors[currentColorIndex];
      Response.Write("<body   bgColor="+ strColor +"></body>");//让网页背景改变颜色。
      //下面的代码演示如何以多种不同的字号显示相同的 HTML 文本。
```

```
            for (int i=0; i<10; i++)
            {
                Response.Write("<font size="+i +"> Hello World! </font>");
            }
            //下面的代码演示以多种不同的颜色显示相同的 HTML 文本。
            String strColor2 = colors[(currentColorIndex+1)%4];
            Response.Write("<font color=" + Color.FromName(strColor2).ToKnownColor () +
                "> 岁月无情增中减，书香有味苦后甜 </font>");
            ViewState["currentColorIndex"] = currentColorIndex.ToString();
        }
        protected void BottonUp_Click(object sender, EventArgs e)
        {
            if (currentColorIndex == 0)
            {
                currentColorIndex = colors.Length - 1;
            }
            else
            {
                currentColorIndex -= 1;
            }
            DisplayColor();
        }
        protected void ButtonDown_Click(object sender, EventArgs e)
        {
            if (currentColorIndex ==(colors .Length -1))
            {
                currentColorIndex = 0;
            }
            else
            {
                currentColorIndex += 1;
            }
            DisplayColor();
        }
    }
```

(5) 保存用户控件。切换到 Default 窗体的【设计】视图，从【解决方案资源管理器】中 2 次拖动用户控件 WebUserControl1 到 Default 窗体上，并添加如图 4-21 所示的文字。



图 4-21　用户控件示例

(6) 按 Ctrl+F5 运行该页。单击 Up、Down 按钮，分析显示结果。

实验 4　用户控件的创建

1. 实验目的

熟悉用户控件的创建技术，掌握用户控件的属性、事件、方法的定义和使用，学会利用自定义用户控件制作导航条和用户登录控件。

2. 实验内容和要求

(1) 新建名字为 ServerControlExperiment 网站。

(2) 在 default.aspx 页面中，添加 1 个 TextBox 控件、2 个 Button 控件、1 个 ListBox 控件，如图 4-22 所示。将 2 个 Button 控件的 Text 属性分别改为“增加”和“删除”。当单击【增加】按钮时，将 TextBox 文本框中的输入值添加到 ListBox 中；当单击【删除】按钮时，删除 ListBox 中当前选定项。

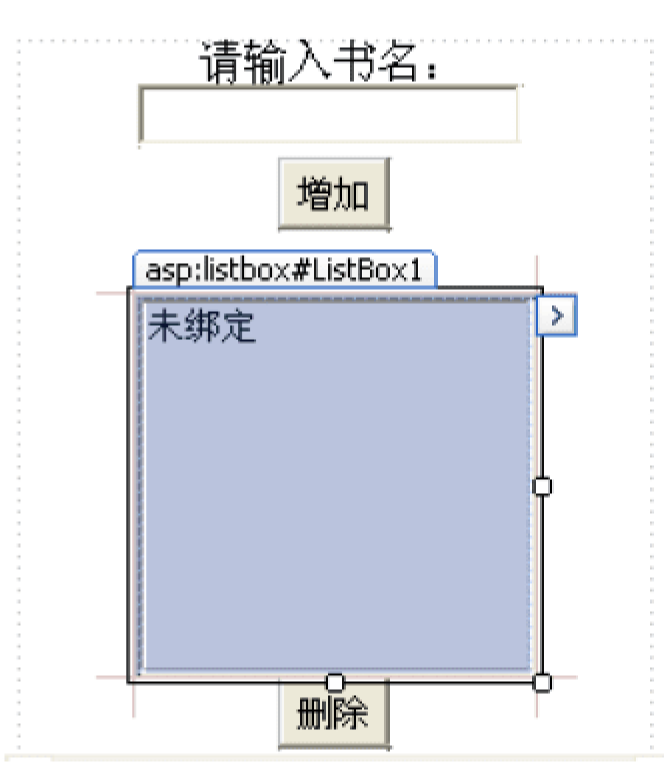


图 4-22　控件

(3) 添加一个网页，要求将 Label 控件、LinkButton 控件、HyperLink 控件放在 Panel 控件中，当单击一组 Button 按钮时改变 Panel 控件的背景色，单击另一组 Button 控件时改变

Panel 控件中文字的大小。单击 LinkButton 和 HyperLink 控件时分别导航到新的网页或网站。单击 RadioButton 控件时隐藏 Panel 控件，单击另一个 RadioButton 控件时显示 Panel 控件。如图 4-23 所示。



图 4-23　控件

(4) 添加一个网页，在 MultiView 控件添加 3 个 View 控件，在每个 View 控件中各添加 Image 控件，再添加 3 个 RadioButton 控件切换各个 View 视图，每个 View 视图显示不同的图像。

(5) 添加一个网页，在页面中添加 AdRotator 控件，链接 XML 广告数据源，实现广告图像显示，当单击广告图像时导航到相应的网页。

(6) 添加一个网页，在页面中添加 CheckBoxList 控件，单击 Button 按钮时将 CheckBoxList 的选项写到 ListBox 中。

(7) 添加一个网页，在页面中添加 RadioButtonList 控件，单击 Button 按钮时将 RadioButtonList 的选项写到 ListBox 中。

(8) 添加一个网页，选择 DropDownList 控件的选项时导航到相应的网站。

(9) 添加一个网页，在页面中添加 TextBox、RequiredFieldValidator 和 CompareValidator 控件，实现 CompareValidator 控件的 Operator 行为的 Equal、GreaterThan 等属性值的验证。

第5章　页面切换与导航

设计一个网站时，用一个网页完成所有功能是不可能的，通常会按照不同的功能将其划分成各自独立的模块进行处理，所以一个网站通常是由很多网页组成的。这样就需要在不同的网页间进行切换，还可能用到网页间的数据传递或数据共享。同时，为了让登录网站的用户顺利地访问到目的网页，还需为网站添加网站导航。用网站导航作为向导，用户可以随时查看到自己所处的位置，各网页之间的关系，因而为用户进一步的浏览网页提供参考。

本章首先介绍几种常用的页面切换方法和数据传递方法，然后介绍与网站导航相关的控件及其基本用法。

本章的学习目标

- 了解页面切换、数据传递和导航的基本知识
- 掌握页面间的切换方法
- 掌握页面间的数据传递方法
- 掌握网站的导航方法

5.1　页面切换

在 ASP.NET Web 应用程序中，有多种页面切换的方法，常用的有以下几种。

(1) 利用超链接切换到其他页面，例如使用<a>标记或者 HyperLink 控件直接链接到其他页面。

(2) 利用 Button、ImageButton 和 LinkButton 控件的 PostBackUrl 属性切换到其他页面。

(3) 使用 Response.Redirect 方法或 Server.Transfer 方法切换到其他页面。

5.1.1　利用超链接切换到其他页面

从一个页面切换到另一个页面最简单的方法就是使用超链接。使用超链接的方法有两种。

1. 一种是使用<a>标记链接到其他页面

例如：

```
<a href="Page2.aspx">进入页面 2</a>
```

2. 另一种是使用 HyperLink 控件链接到其他页面

例如：

```
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/page2.aspx">进入页面 2
</asp:HyperLink>
```

【例 5-1】演示如何利用 HyperLink 控件链接到其他页面。

(1) 运行 VWD 2008，新建一个名为 HyperLink_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加一个名为 Page1.aspx 的网页，添加一个名为 Page2.aspx 的网页。

(3) 切换到 Page1.aspx 的【设计】视图，向页面中拖放一个 HyperLink 控件，使用默认的控件名称，然后设置 HyperLink 控件的 NavigateUrl 属性。在 HyperLink 控件的【属性】面板中，单击 NavigateUrl 右侧的按钮，在弹出的【选择 URL】对话框中选择 Page2.aspx，单击【确定】按钮，使其属性值为 Page2.aspx，如图 5-1 所示。并设置 HyperLink 控件的 Text 属性值为“进入页面 2”。

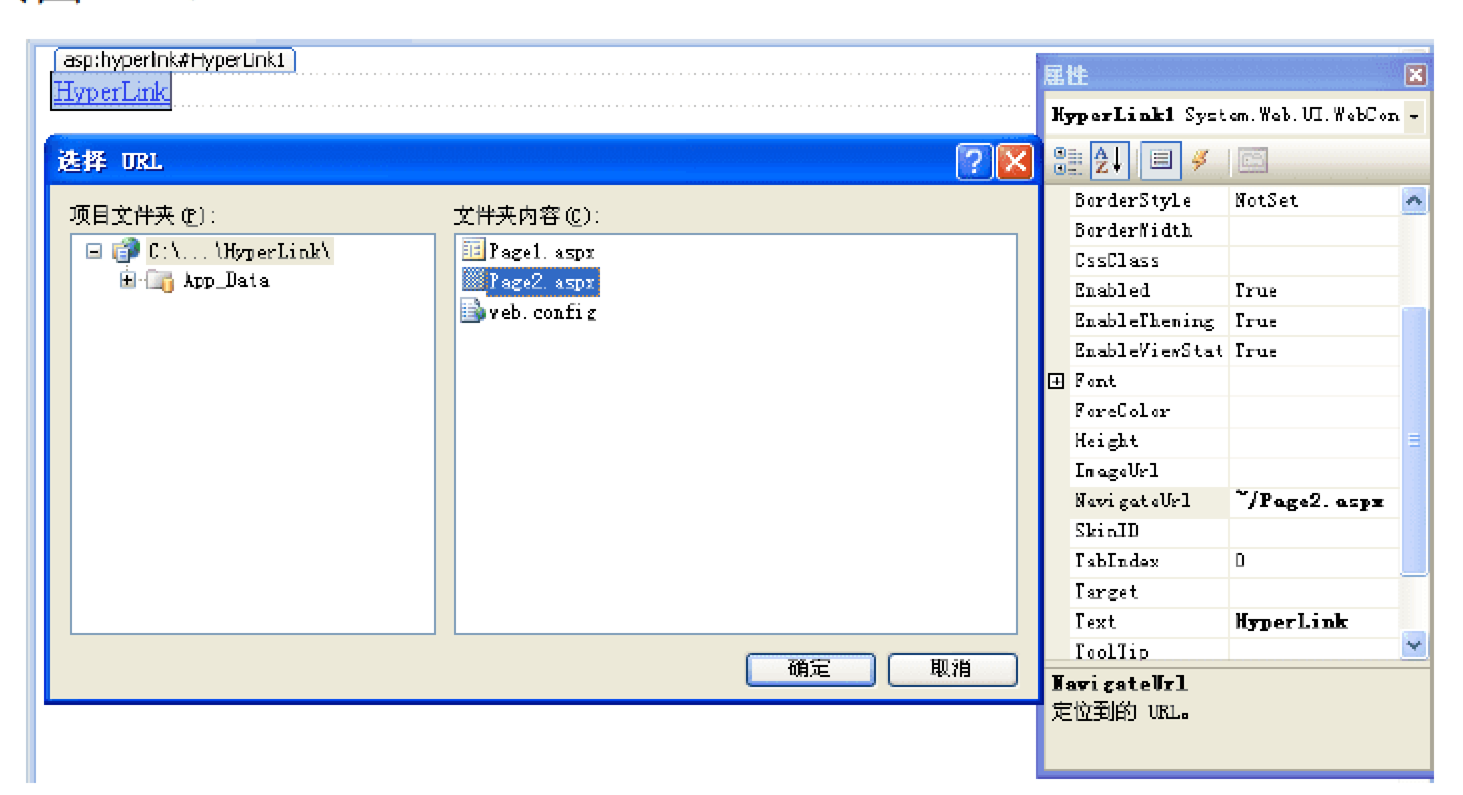


图 5-1　设置 HyperLink 的 NavigateUrl 属性

(4) 切换到 Page1.aspx 网页，按 F5 键调试运行。

使用 HyperLink 控件链接到其他页面的链接效果与使用<a>标记链接到其他网页的效果相同。但使用 HyperLink 控件的好处是可以在服务器端动态地设置 NavigateUrl 属性，例如：

```
Protected void Page_Load(object sender, EventArgs e)
{
 If  <判断条件>
   {
    HyperLink1.NavigateUrl="NewPage.aspx";        *判断条件为“真”时，执行此语句
   }
 else
   {
    HyperLink1.NavigateUrl="";                    *判断条件为“假”时，执行此语句
   }
}
```

超链接方式的特点是，当用户单击超链接时，客户端浏览器会直接请求链接的目标网页，

因此服务器不会将目标网页与链接前的网页(源网页)进行关联，也就是说，服务器不会将与源网页有关的任何信息传递到目标网页，除非在目标网页的 URL 上指定查询字符串。

在同一个 Web 应用程序中，如果使用超链接切换到另一个网页，可以使用会话状态(Session)来共享源网页和目标网页中的信息。

5.1.2　利用按钮的 PostBackUrl 属性切换到其他页面

在 Button、LinkButton 和 ImageButton 控件中，有一个 PostBackUrl 属性，可以利用该属性切换到其他页面，这种切换方式称为跨页发送。

跨页发送与超链接都是通过用户操作来切换到其他页面。但是，在跨页发送中，服务器会将源网页上控件的值发送到目标网页。如果源网页和目标网页属于同一个 Web 应用程序，目标网页还可以访问源网页的公共属性。

利用按钮 Button、LinkButton 和 ImageButton 控件的 PostBackUrl 属性切换到其他页面的操作过程与利用超链接切换到其他页面的操作过程相似。

【例 5-2】演示如何利用 Button、LinkButton 和 ImageButton 控件的 PostBackUrl 属性链接到其他页面。

(1) 运行 VWD 2008，新建一个名为 PostBackUrl_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加一个名为 Page1.aspx 的网页，再分别添加名为 Page2.aspx、Page3.aspx、Page4.aspx 的网页，并分别在 Page2.aspx、Page3.aspx、Page4.aspx 的网页中添加文字“欢迎来到页面 2!”、“欢迎来到页面 3!”和“欢迎来到页面 4!”，以便运行时将不同页面区分开。

(3) 切换到 Page1.aspx 的【设计】视图，向页面中拖放 1 个 Button 控件，1 个 LinkButton 控件和 1 个 ImageButton 控件，均使用默认的控件名称。然后分别设置 Button、LinkButton 和 ImageButton 控件的 PostBackUrl 属性。在 Button 控件的【属性】面板中，单击 PostBackUrl 右侧的按钮，在弹出的【选择 URL】对话框中选择 Page2.aspx，单击【确定】按钮，使其属性值为 Page2.aspx，如图 5-2 所示。并设置 Button 控件的 Text 属性值为“单击 Button 转到页面 2”。类似地，设置 LinkButton 控件的 PostBackUrl 属性值为 Page3.aspx，Text 属性值为“单击 LinkButton 转到页面 3”，设置 ImageButton 控件的 PostBackUrl 属性值为 Page4.aspx，Text 属性值为“单击 ImageButton 转到页面 4”。

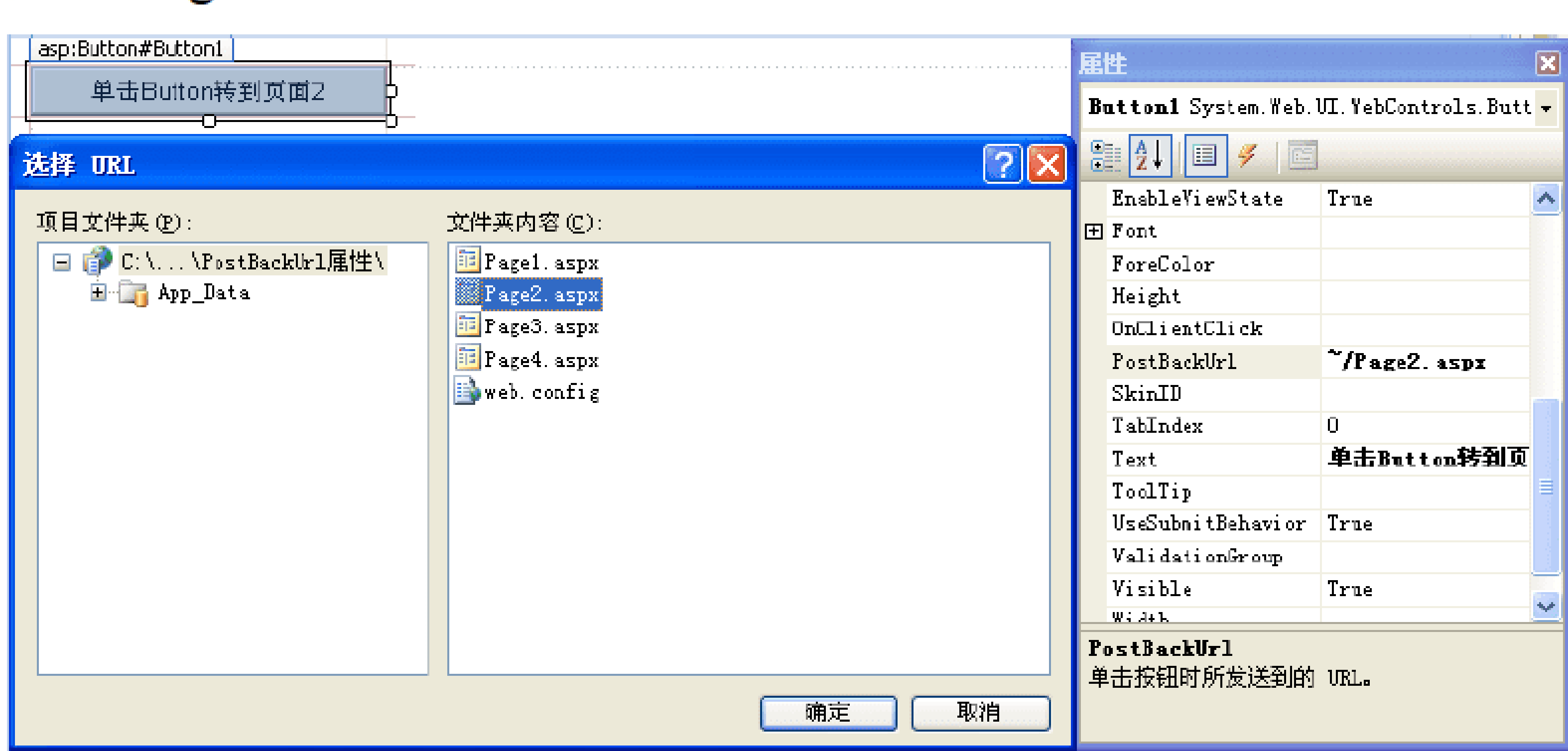


图 5-2　设置 Button 控件的 PostBackUrl 属性

(4) 切换到 Page1.aspx 网页，按<F5>键调试运行。

5.1.3 使用 Response.Redirect 或 Server.Transfer 切换到其他页面

虽然利用超链接和 Button、ImageButton、LinkButton 控件的 PostBackUrl 属性能够从一个页面切换到其他页面，但这种切换方式是完全由用户控制的。在某些情况下，可能需要用服务器代码来控制什么时候切换到另一个页面，此时就可以利用 Response 对象的 Redirect 方法或者 Server 对象的 Transfer 方法。

利用 Response 对象的 Redirect 方法与利用 Server 对象的 Transfer 方法进行页面切换的方式基本相似，但也有一些区别。

1. Response.Redirect

Response.Redirect 方法会使浏览器链接到一个指定的 URL。当服务器调用 Response.Redirect 方法时，该方法会创建一个应答，应答头中指出目标网页已经改变，同时指出新目标的 URL。客户端浏览器从服务器收到该应答，再利用应答头中的信息发出一个对新页面 URL 的请求。

这就是说，使用 Response.Redirect 方法时重定向操作发生在客户端，共涉及两次与服务器的通信(两个来回)：第一次是对原始页面的请求，得到一个应答；第二次是请求应答中声明的新页面，得到重定向之后的页面。

2. Server.Transfer

Server.Transfer 方法直接在服务器端把执行流程从当前的页面转到同一个应用程序的另一个页面。调用 Server.Transfer 方法时，当前页面终止执行，然后将执行流程转入另一个页面，但新的页面仍使用前一页面创建的应答流。

用 Server.Transfer 方法从一个页面切换到另一个页面，浏览器中的 URL 不会改变，因为重定向完全在服务器端进行，浏览器根本不知道服务器已经执行了一次页面切换。

Server.Transfer 方法有两种重载的形式，一种是只有一个参数，该参数指出新页面的 URL，这种情况下，服务器不会把页面中的表单数据或查询字符串从一个页面传递到另一个页面。此种用法在后面的【例 5-4】中有所体现。另一种重载形式包含两个参数，第一个参数仍是新页面的 URL，而第二个参数则指出是否保留原页面的表单数据和查询字符串以供新页面查询。此种用法在后面的【例 5-3】中有所体现。

3. 两者之间的区别

利用 Response 对象的 Redirect 方法与利用 Server 对象的 Transfer 方法进行页面切换的方式除了上面的不同之外，还有以下区别。

(1) Response.Redirect 方法不限于当前应用程序，也不限于.aspx 网页，利用它可以重定向到任何页面，包括其他网站中的页面，也可以重定向到任何文件，比如.rar 文件等，如果不是网页，浏览器会显示相应的对话框，供用户选择是直接打开还是保存到磁盘上。

Server.Transfer 方法则不同，该方法只能切换到同一个应用程序的.aspx 网页。

(2) 对 Response.Redirect 方法来说，切换到另一个页面之后，浏览器的地址栏将显示新页面的 URL，对于传递不希望用户看到的字符串信息，这种方法就不适合了。

Server.Transfer 方法则可以传递不希望用户看到的字符串信息，当用户切换到新的页面后，浏览器的地址栏仍然显示原来的地址。

5.1.4　页面间的数据传递

在实际应用中，一个页面可能需要访问或者使用另一个页面中的数据，下面是几种常用的页面间数据传递的方法。

1. 使用 QueryString 传递数据

在使用 QueryString 传递数据时，把要传递的数据参数添加到所要切换到的另一个页面的 URL 后。在要传递的第一个数据参数前用？将其和目标页面的名称分隔开，参数之间用&分隔，每个参数的形式为“变量名=值”。而在另一个页面中使用 Request.QueryString[stringName]接收传递的参数。

【例 5-3】演示如何直接在页面名称后附带参数传递数据。

(1) 运行 VWD 2008，新建一个名为 QueryString_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加 1 个名为 FirstPage.aspx 的网页。在 FirstPage.aspx 中，添加 2 个 TextBox 控件，1 个 Button 控件，均使用默认的控件名称。分别在 2 个 TextBox 控件前添加文字“学号：”和“姓名：”，设置 Button 控件的 Text 属性值为“提交”。然后添加 Button 控件的 Click 事件代码。

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Server.Transfer(string.Format("SecondPage.aspx?学号={0}&姓名={1}",
        Server.UrlEncode(TextBox1.Text), Server.UrlEncode(TextBox2.Text)));
    }
```

(3) 在应用程序中添加 1 个名为 SecondPage.aspx 的网页。在 SecondPage.aspx 中，添加 2 个 Label 控件，均使用默认的控件名称，然后在 SecondPage.aspx 的 Page_load 事件中添加下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "学号：" + Server.UrlDecode(Request.QueryString["学号"]);
        Label2.Text = "姓名：" + Server.UrlDecode(Request.QueryString["姓名"]);
    }
```

(4) 切换到 FirstPage.aspx 网页，按 F5 键调试运行。运行效果如图 5-3 和图 5-4 所示。

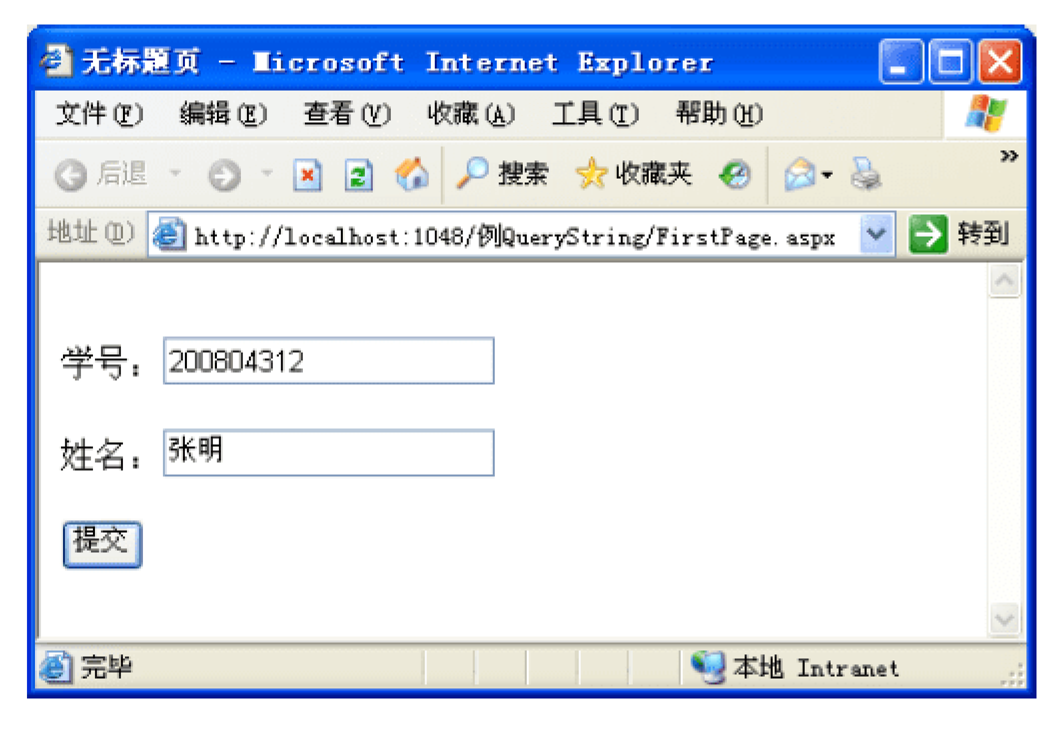


图 5-3　在第一个页面传递数据的效果

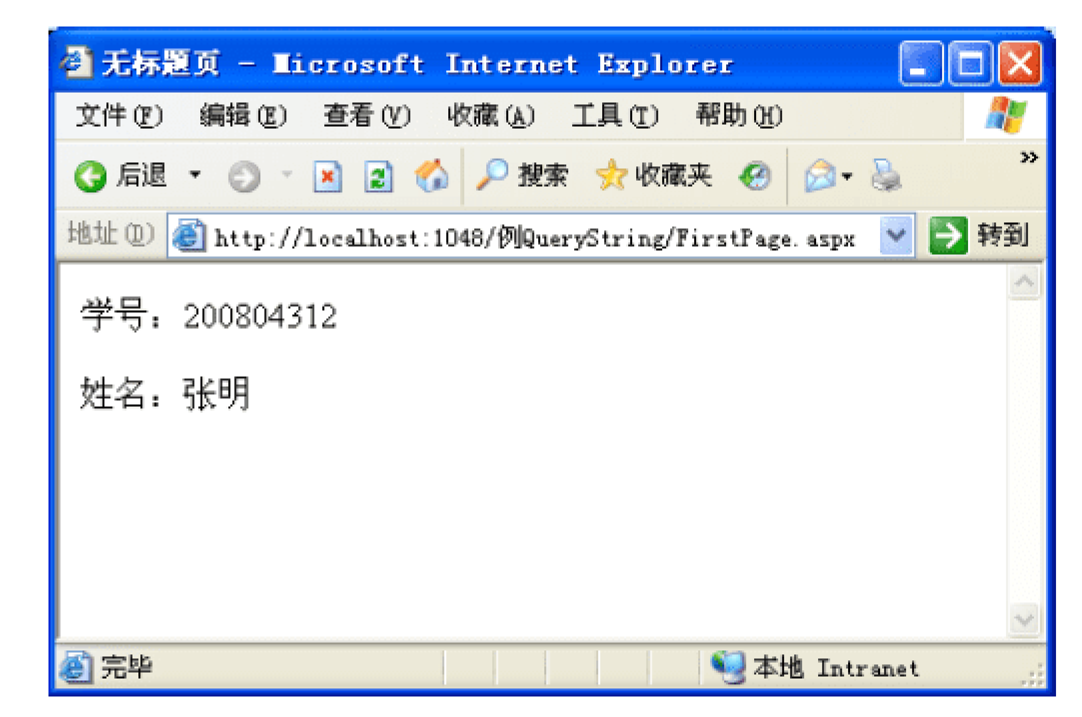


图 5-4　在第二个页面接收数据的效果

在使用 QueryString 传递数据的方法中，使用&作为参数之间分隔符。为了保证接收方能得到正确的结果，需要在传递数据参数前使用 Server.UrlEncode 方法对字符串进行编码，接收方接收后再调用 Server.UrlDecode 方法对相应的字符串进行解码。UrlEncode 将指定的代码以 URL 格式进行编码。UrlDecode 将 URL 格式代码进行解码。

这里要注意，如果直接运行 SecondPage.aspx，由于没有接收到任何传递的参数，Request.QueryString[stringName]将返回一个空字符串。

2．利用 Session 对象传递数据。

由于每个用户都可以有自己的 Session，而创建的 Session 对象并不局限于某一个页面，所以可以使用 Session 对象在多个页面之间共享数据。

【例 5-4】演示如何利用 Session 对象传递数据。

(1) 运行 VWD 2008，新建一个名为 Session_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加 1 个名为 FirstPage.aspx 的网页，在 FirstPage.aspx 中，添加 2 个 TextBox 控件，1 个 Button 控件，均使用默认的控件名称。分别在 2 个 TextBox 控件前添加文字“学号：”和“姓名：”，设置 Button 控件的 Text 属性值为“提交”。然后添加 Button 控件的 Click 事件代码。

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Session["学号"] = TextBox1.Text;
        Session["姓名"] = TextBox2.Text;
        Server.Transfer("SecondPage.aspx");
    }
```

(3) 在应用程序中添加 1 个名为 SecondPage.aspx 的网页，在 SecondPage.aspx 中，添加 2 个 Label 控件，均使用默认的控件名称，然后在 SecondPage.aspx 的 Page_Load 事件中添加下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Session ["学号"]!=null)
        {
```

```
            Label1.Text = "学号：" + Session["学号"].ToString();
        }
        if (Session ["姓名"]!=null)
        {
            Label2.Text = "姓名：" + Session["姓名"].ToString();
        }
    }
```

(4) 切换到 FirstPage.aspx 网页，按 F5 键调试运行。运行效果与【例 5-3】相似。

需要注意的是，这种方法会占用服务器的内存资源，并且每个用户的 Session 信息将一直存储在内存中直到 Session 过期，从而导致系统开销较大。

下面介绍利用 PreviousPage 属性获取上一页数据的方法，此方法适合用于页面间传递较多的信息时使用。

3. 利用 PreviousPage 属性获取数据

如果两个 ASP.NET 页面属于同一个应用程序，当在源页面中利用 Server.transfer 方法或者利用按钮控件的 PostBackUrl 属性切换到目标页面时，目标页面可以使用 PreviousPage 属性来获取源页面中的控件值或公共属性。

如果要获取源页面中的公共属性，则需要在目标页面的源代码中添加如下代码：

```
<%@ PreviousPageType VirtualPath="~/SourcePage.aspx" %>
```

但是，如果目标页面不需要获取源页面的公共属性，而仅仅需要获取源页面中的控件值，则不需要添加这行代码。

【例 5-5】演示如何利用 PreviousPage 属性获取源页面的控件值。

(1) 运行 VWD 2008，新建一个名为 PreviousPage_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加 1 个名为 FirstPage.aspx 的网页，在 FirstPage.aspx 中，添加 2 个 TextBox 控件，1 个 Button 控件，均使用默认的控件名称。 分别在 2 个 TextBox 控件前添加文字“学号：”和“姓名：”，设置 Button 控件的 Text 属性值为“提交”。然后添加 Button 控件的 Click 事件代码。

```
sxprotected void Button1_Click(object sender, EventArgs e)
    {
        Server.Transfer("SecondPage.aspx");
     }
```

(3) 在应用程序中添加 1 个名为 SecondPage.aspx 的网页，在 SecondPage.aspx 中，添加 2 个 Label 控件，均使用默认的控件名称，然后在 SecondPage.aspx 的 Page_load 事件中添加下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (PreviousPage != null)
```

```
        {
            TextBox textbox1 = (TextBox)PreviousPage.FindControl("TextBox1");
            TextBox textbox2 = (TextBox)PreviousPage.FindControl("TextBox2");
            Label1.Text = "学号："+Textbox1.Text;
            Label2.Text = "姓名："+Textbox2.Text;
        }
    }
```

(4) 切换到 FirstPage.aspx 网页，按 F5 键调试运行。运行效果与【例 5-3】相似。

在 SecondPage.aspx 网页中，使用了 PreviousPage.FindControl 获取 FirstPage.aspx 网页中的控件值。这种方法与利用 Session 对象传递数据相比，不需要一直占用服务器资源，适合于页面间传递较多的信息时使用，使页面间的数据传递更加灵活方便。

【例 5-5】演示了如何利用 PreviousPage 属性获取源页面的控件值。如果要在 SecondPage.aspx 网页中访问 FirstPage.aspx 网页中的公共属性，可以按照下面的步骤进行。

(1) 在 FirstPage.aspx 中定义属性。例如：

```
 public string Number
{
    get {return TextBox1.Text;}
 }
public string UserName
{
    get {return TextBox2.Text;}
 }
protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("SecondPage.aspx");
}
```

(2) 在 SecondPage.aspx 的【源】视图中添加以下代码：

```
<%@PreviousPageType VirtualPath="~/FirstPage.aspx"%>
```

然后就可以在 SecondPage.aspx 的后台代码中使用 FirstPage.aspx 中定义的 Number 和 UserName 属性了。例如：

```
Protected void Page_load(object sender, EventArgs e)
{
    If(PreviousPage!=null)
    {
         Label1.Text="学号："+PreviousPage.Number;
         Label2.Text="姓名："+PreviousPage.UserName;
    }
}
```

5.2　网站导航

在最细微的层次上，网站不过是由很多网页组成的集合。然而，通常这些网页都是逻辑上相关联且以某种方式分类的。例如，一个网上商店可以按产品分类组织网站，如书籍、CD、DVD 等。这些部分又可以分别按各自的种类分类，如书籍可以分为计算机类书籍、经济类书籍等。这样可以使网站具有清晰的逻辑结构。将网页分组成不同的逻辑类别称为网站的结构。

定义网站的结构后，大多数 Web 开发人员将创建网站导航。网站导航是用于帮助用户浏览网站的用户界面元素集合。常见的导航元素包括面包条、菜单和树视图。这些用户界面元素常用于完成两种任务，一是让用户知道自己在所访问网站中的位置，二是让用户更容易、更快速地跳转到网站的其他部分。

5.2.1　ASP.NET 网站导航功能概述

网站导航主要提供了如下功能：

(1) 使用站点地图描述网站的逻辑结构。添加或移除页面时，开发人员可以简单地通过修改站点地图来管理页面导航。

(2) 提供导航控件，在页面上显示导航菜单。导航菜单以站点地图为基础。

(3) 可以以代码方式使用 ASP.NET 网站导航，以创建自定义导航控件或修改在导航菜单中显示的信息的位置。

ASP.NET 3.5 提供了 3 种导航 Web 控件。

(1) SiteMapPath：这个 Web 控件提供一个面包条(breadcrumb)，它是一行文本，显示用户当前在网站结构中的位置。例如，在网上书店中，如果用户浏览到《Visual C++》，面包条可能类似于“主页->计算机->编程类->Visual C++”，其中每部分(如主页，计算机等)都显示为返回到前一部分的链接。面包条能够让用户快速地查看当前在网站中的位置，并沿逻辑层次结构向上导航。图 5-7 显示了运行中的 SiteMapPath 控件。

(2) Menu：这个 Web 控件提供网站结构的层次视图。对于学校的网站，顶层菜单将包含主类别(如学校介绍、机构设置、新闻等)，每个菜单项又可以有各自的子菜单，显示各自的子类别。关于运行中的 Menu 控件可参见图 5-10。

(3) TreeView：树视图提供了与菜单相同的数据，唯一的区别是显示数据的方式。树视图显示的是可展开或可折叠的树，而菜单(Menu)由菜单项和子菜单组成。图 5-11 显示了通过浏览器查看时的 TreeView 控件。

一般情况下，开发人员利用站点地图和 SiteMapPath 控件实现自动导航，利用 Menu 控件或者 TreeView 控件实现自定义导航。

5.2.2　利用站点地图和 SiteMapPath 控件实现网站导航

要使用 SiteMapPath 导航控件，首先需要使用站点地图定义网站的结构，创建站点地图

文件。然后使用 SiteMapPath 控件实现网站导航。

要创建站点地图，可遵循在应用程序中添加 ASP.NET 网页的步骤。也就是在【解决方案资源管理器】中右击应用程序名称，在弹出菜单中选择【添加新项】，然后在弹出的【添加新项】对话框中，选择【站点地图】选项(参见图 5-6 所示)并单击【添加】按钮。这将为应用程序添加一个名为 Web.sitemap 的站点地图。

注意：

添加站点地图到应用程序中时，需要将站点地图放在 Web 应用程序的根目录下，并保持其文件为 Web.sitemap。如果将该文件放在另一个文件夹中或选择不同的文件名，SiteMapPath 导航控件将不能找到站点地图，就不能知道网站的结构，因为默认情况下 SiteMapPath 导航控件在根目录下寻找名为 Web.sitemap 的文件。

添加站点地图后，在【解决方案资源管理器】中双击 Web.sitemap 文件，打开这个文件，将显示默认情况下站点地图中的标记，程序清单如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
        <siteMapNode url="" title=""   description="">
            <siteMapNode url="" title=""   description="" />
            <siteMapNode url="" title=""   description="" />
        </siteMapNode>
</siteMap>
```

站点地图是指描述网站逻辑结构的 XML 文件，该文件的扩展名为.sitemap。这个 XML 文件包含网站的逻辑结构。要定义网站的结构，需要手工编辑这个文件。

注意：

内部没有内容的 XML 元素可以采用两种形式的结束标签：一种是冗余方式，如<myTag attribute="value"…></myTag>，另一种是使用简洁方法，如<myTag attribute="value"…/>。

定义好站点地图以后，就可以使用 SiteMapPath 控件显示导航路径，也就是显示当前页面在网站中的位置。只需要将该控件拖放到站点地图中包含的.aspx 页面上，它就会自动实现导航，不需要开发者编写任何代码。

注意：

只有包含在站点地图中的网页才能被 SiteMapPath 控件导航；如果将 SiteMapPath 控件放置在站点地图中未列出的网页中，该控件将不会显示任何信息。

SiteMapPath 控件像大多数 Web 控件一样，也有很多可用于定制其外观的属性。表 5-1 所示为 SiteMapPath 控件的常用属性。

表 5-1　SiteMapPath 控件的常用属性

属　性　名	说　　明
CurrentNodeStyle	定义当前节点的样式，包括字体、颜色、样式等
NodeStyle	定义导航路径上所有节点的样式
ParentLevelsDisplayed	指定在导航路径上显示的相对于当前节点的父节点层数。默认值为-1，表示父级别数没有限制
PathDirection	指定导航路径上各节点的显示顺序。默认值为 RootToCurrent，即按从左到右的顺序显示从根节点到当前节点的路径。另一选项为 CurrentToRoot，即按相反的顺序显示导航路径
PathSeparator	指定导航路径中节点之间分隔符。默认值为>，也可自定义为其他符号
PathSeparatorStyle	定义分隔符的样式
RenderCurrentNodeAsLink	是否将导航路径上当前页名称显示为超链接。默认值为 false
RootNodeStyle	定义根节点的样式
ShowToolTips	当鼠标悬停于导航路径的某个节点时，是否显示相应的工具提示信息。默认值为 true，即当鼠标悬停于某节点上时，显示该节点在站点地图中定义的 Description 属性值

下面通过具体例子演示如何利用站点地图和 SiteMapPath 控件实现自动导航。

【例 5-6】创建如图 5-5 所示的站点地图，然后利用 SiteMapPath 控件实现自动导航。

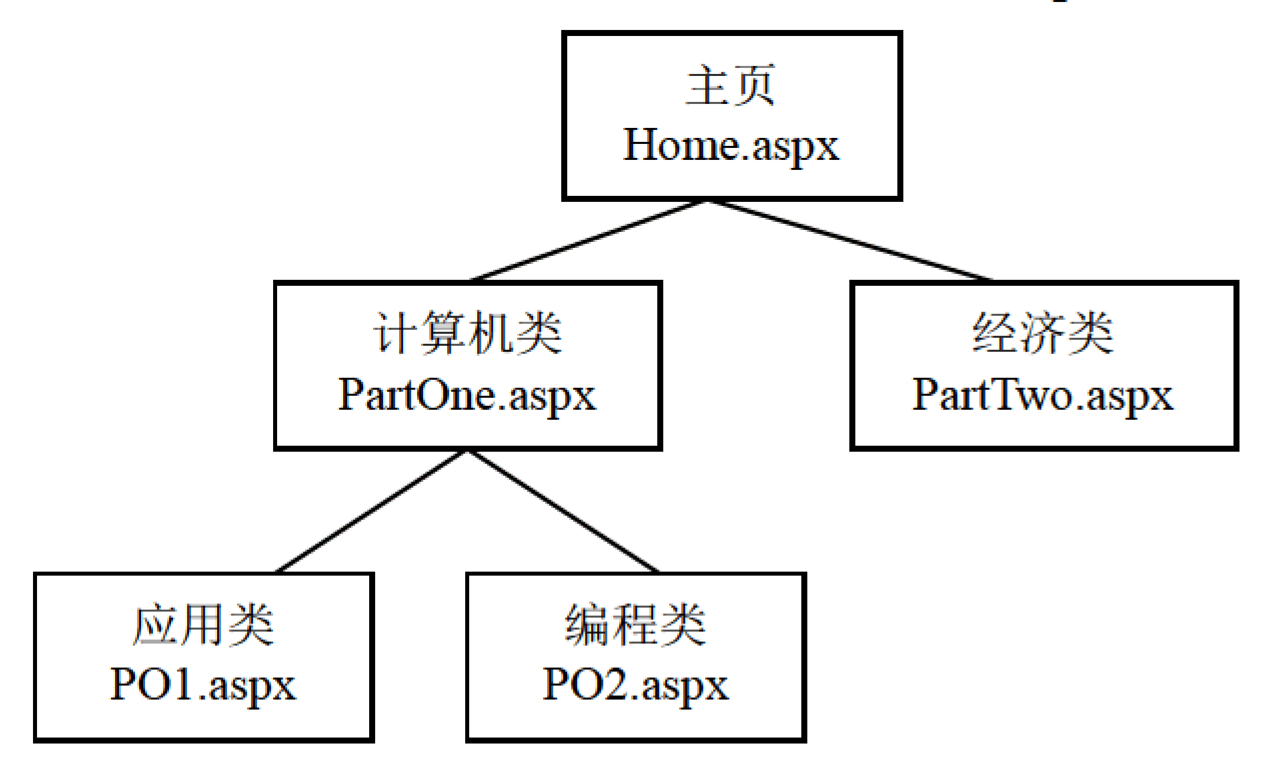


图 5-5　网上书店网站的逻辑结构

(1) 运行 VWD 2008，新建一个名为 SiteMapPath_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中添加一个名为 Web.sitemap 的站点地图。如图 5-6 所示。



图 5-6　创建站点地图

(3) 将 Web.sitemap 文件中的内容改为如下形式：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/Home.aspx" title="主页"  description="Home">
        <siteMapNode url="~/PartOne.aspx" title="计算机类"  description="单击此链接转到计算机
          类" >
            <siteMapNode url="~/PO1.aspx" title="应用类"  description="单击此链接转到应用类
              " />
            <siteMapNode url="~/PO2.aspx" title="编程类"  description="单击此链接转到编程类
              " />
        </siteMapNode>
        <siteMapNode url="~/PartTwo.aspx" title="经济类"  description="单击此链接转到经济类" >
        </siteMapNode>
    </siteMapNode>
</siteMap>
```

注意：

站点地图文件中只能有一个根节点，即位于<sitemap>下方的第一个<siteMapNode>元素中的 Home.aspx 页面。在根节点下可以嵌套任意多个子节点，子节点仍然用<siteMapNode>定义。

在每个节点的定义中，title 实现在导航控件中显示指定页面的名称，description 实现鼠标悬停于导航控件的某个节点时所要显示的提示信息，url 实现指定节点对应的页面路径。

(4) 保存文件，完成站点地图的设计。

定义了站点地图之后，就可以在导航控件中轻松实现导航功能。

(5) 在【解决方案资源管理器】中，分别添加名为 Home.aspx、PartOne.aspx、PartTwo.aspx、PO1.aspx 和 PO2.aspx 网页。

(6) 切换到 PartOne.aspx 的【设计】视图，向页面中拖放 1 个 SiteMapPath 控件，即可以

看到该页面相对应于 Home.aspx 的导航路径，如图 5-7(a)所示。

(7) 切换到 PO2.aspx 的【设计】视图，向页面中拖放 1 个 SiteMapPath 控件，即可看到该页面相对应于 Home.aspx 和 PartOne.aspx 的导航路径，如图 5-7 (b)所示。



(a) 拖放到 PartOne.aspx 的效果　　(b) 拖放到 PO2.aspx 的效果

图 5-7　将 SiteMapPath 控件拖放到页面后看到的效果

可见，利用站点地图和 SiteMapPath 控件实现自动导航非常方便。如果不希望采用这种方式导航，也可以利用 Menu 控件或者 TreeView 控件实现自定义导航功能。

5.2.3　利用 Menu 控件实现自定义导航

Menu 控件主要用于创建一个菜单，让用户快速选择不同页面，从而完成导航功能。该控件可以包含一个主菜单和多个子菜单。菜单有静态和动态两种显示模式。静态显示模式是指定义的菜单始终完全显示，动态显示模式指需要用户将鼠标停留在菜单项上时才显示子菜单。

Menu 控件的常用属性如表 5-2 所示。Menu 控件的属性很多，这里不逐一介绍。

表 5-2　Menu 控件的常用属性

属 性 名	说　　明
DynamicEnableDefaultPopOutImage StaticEnableDefaultPopOutImage	是否在菜单各项之间显示分隔图像。默认值为 true
DynamicPopOutImageUrl StaticPopOutImageUrl	设置菜单中自定义分隔图像的 URL
DynamicBottomSeparatorImageUrl StaticBottomSeparatorImageUrl	指定在菜单项下方显示图像的 URL。默认值为空字符串("")，即菜单项下方不显示任何图像
DynamicTopSeparatorImageUrl StaticTopSeparatorImageUrl	指定在菜单项上方显示图像的 URL。默认值为空字符串("")，即菜单项上方不显示任何图像
DynamicHorizontalOffset StaticHorizontalOffset	指定菜单相对于其父菜单的水平距离，单位是像素，默认值为 0。该属性值可正可负，为负值时，各菜单之间的距离会缩小
DynamicVerticalOffset StaticVerticalOffset	指定菜单相对于其父菜单项的垂直距离
MaximumDynamicDisplayLevels	设置动态菜单的最大层数。默认值为 3
Orientation	设置菜单的展开方向。有 Horizontal 和 Vertical 两个选项，默认值为 Vertical，即垂直方向

Menu 控件的用法非常灵活，设计者可以利用它定义各种菜单样式，实现类似于 Windows 窗口菜单的功能。

下面通过一个具体的例子演示如何利用 Menu 控件实现自定义导航。

【例 5-7】假定网站的结构如图 5-8 所示，然后利用 Menu 控件在网页中添加一个菜单，实现自定义导航功能。

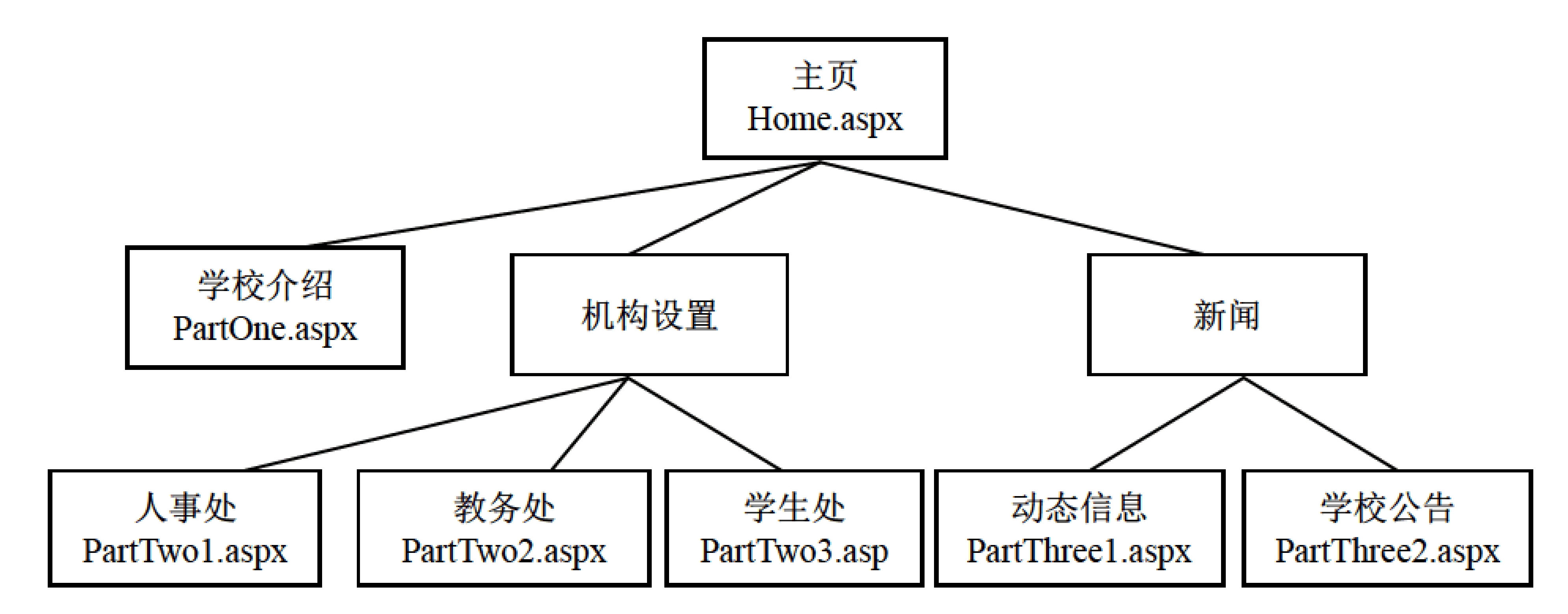


图 5-8　学校网站的逻辑结构

设计步骤如下。

(1) 运行 VWD 2008，新建一个名为 Menu_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中分别添加名为 PartOne.aspx、PartTwo1.aspx、PartTwo2.aspx、PartTwo3.aspx、PartThree1.aspx 和 PartThree2.aspx 网页。

(3) 在应用程序中添加一个名为 MenuExample.aspx 的网页，然后切换到【设计】视图，向页面拖放一个 Menu 控件。

(4) 将 Menu 控件的 Orientation 属性设置为 Horizontal，以便使其横向排列。

(5) 单击 Menu 控件右上方的小三角符号，选择【编辑菜单项】，在弹出的【菜单项编辑器】对话框中，输入各级菜单项，如图 5-9 所示。

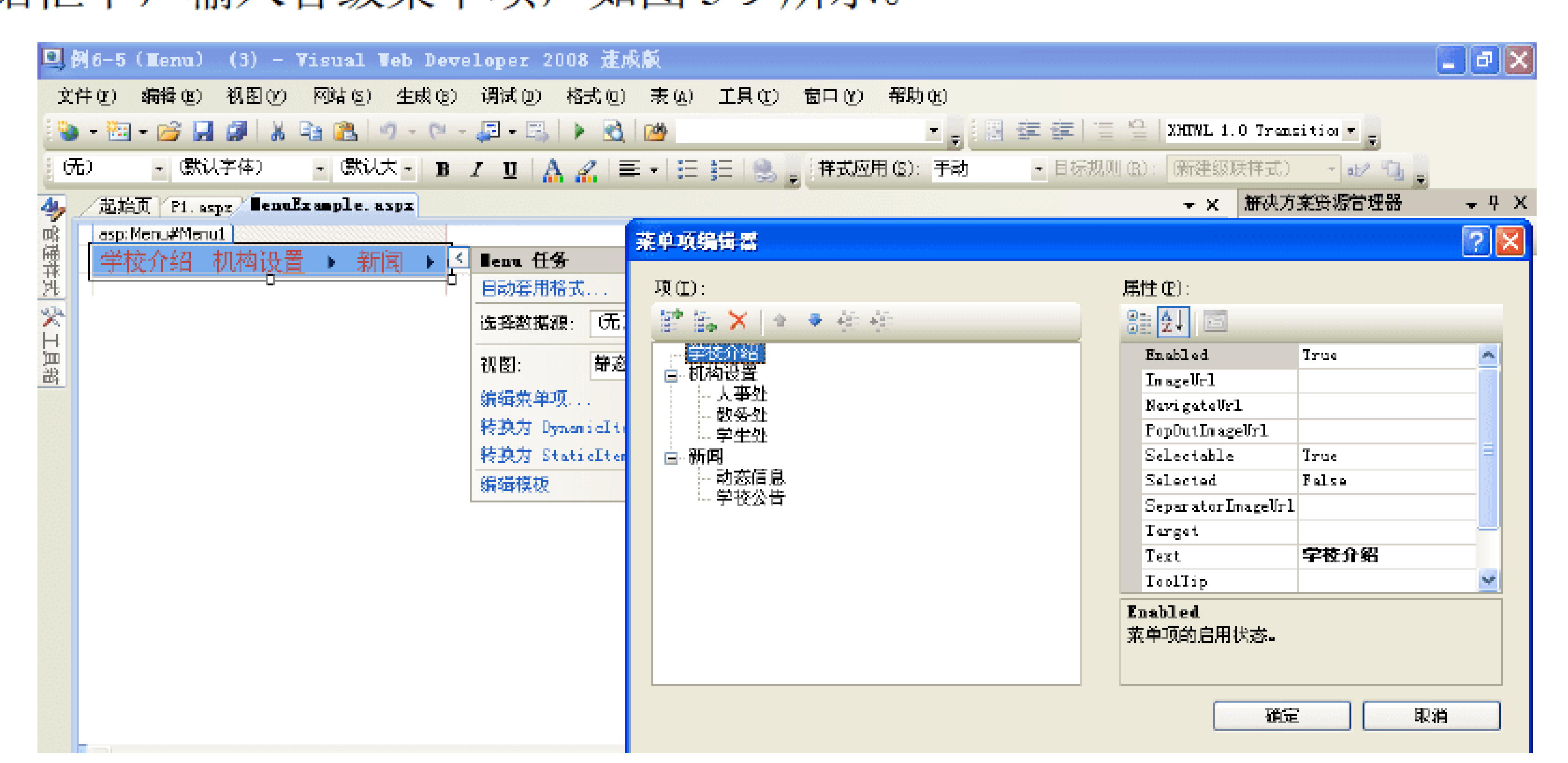


图 5-9　在【菜单编辑器】中编辑菜单

(6) 在【菜单编辑器】窗口右侧的属性选项中，利用 NavigateUrl 属性设置各菜单项链接的网页，全部设置完成后，单击【确定】按钮。

(7) 切换到 MenuExample.aspx 的【源】视图，将<body>和</body>之间的部分改为如下

内容：

```
<asp:Menu ID="Menu1" runat="server"
            EnableViewState="False"
            DynamicHorizontalOffset="2"
            DynamicVerticalOffset="5"
            Target="_blank"
            Font-Names="Verdana"
            Font-Size="Medium"
            ForeColor="#FF3300"
            BackColor="#99CCFF"
            StaticSubMenuIndent="10px" Orientation="Horizontal" >
           <StaticHoverStyle BackColor="#FFCCCC" ForeColor="white" />
           <StaticSelectedStyle    BackColor="#FFCC66"/>
           <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding ="2px" />
           <DynamicHoverStyle BackColor ="#990000" ForeColor ="White" />
           <DynamicMenuStyle BackColor ="#FFFBD6"   BorderColor ="#00C0C0"
            BorderStyle ="Solid" BorderWidth ="1px"
            HorizontalPadding ="10px" VerticalPadding ="2px" />
           <DynamicSelectedStyle BackColor ="#FFCC66" />
           <DynamicMenuItemStyle HorizontalPadding ="5px" ItemSpacing= "2px" />
            <Items>
              <asp:MenuItem Text="学校介绍" NavigateUrl="~/PartOne.aspx" >
              </asp:MenuItem>
              <asp:MenuItem Text="机构设置" Value="机构设置" >
                 <asp:MenuItem Text="人事处" NavigateUrl="~/PartTwo1.aspx" Value="人事处
                   " ></asp:MenuItem>
                 <asp:MenuItem Text="教务处" NavigateUrl="~/PartTwo2.aspx" Value="教务处
                   " ></asp:MenuItem>
                 <asp:MenuItem Text="学生处" NavigateUrl="~/PartTwo3.aspx" Value="学生处
                   "></asp:MenuItem>
               </asp:MenuItem>
               <asp:MenuItem Text="新闻" Value="新闻" >
                    <asp:MenuItem Text="动态信息" NavigateUrl="~/PartThree1.aspx" Value="动
                      态信息" ></asp:MenuItem>
                    <asp:MenuItem Text="学校公告" NavigateUrl="~/PartThree2.aspx" Value="学
                      校公告" ></asp:MenuItem>
               </asp:MenuItem>
            </Items>
      </asp:Menu>
```

当然，也可以在【设计】视图下设置 Menu 控件的各种属性得到上面的代码。

(8) 为了便于区别本例子中的各个网页，分别在 PartOne.aspx、PartTwo1.aspx、PartTwo2.aspx、PartTwo3.aspx、PartThree1.aspx 和 PartThree2.aspx 网页中添加文字“欢迎了

解学校，以下将对学校进行介绍”、“欢迎来到人事处网页！”、“欢迎来到教务处网页！”、“欢迎来到学生处网页！”、“欢迎了解动态信息！”和“欢迎了解学校公告！”。

(9) 切换到 MenuExample.aspx 网页，按 F5 键调试运行，效果如图 5-10 所示。

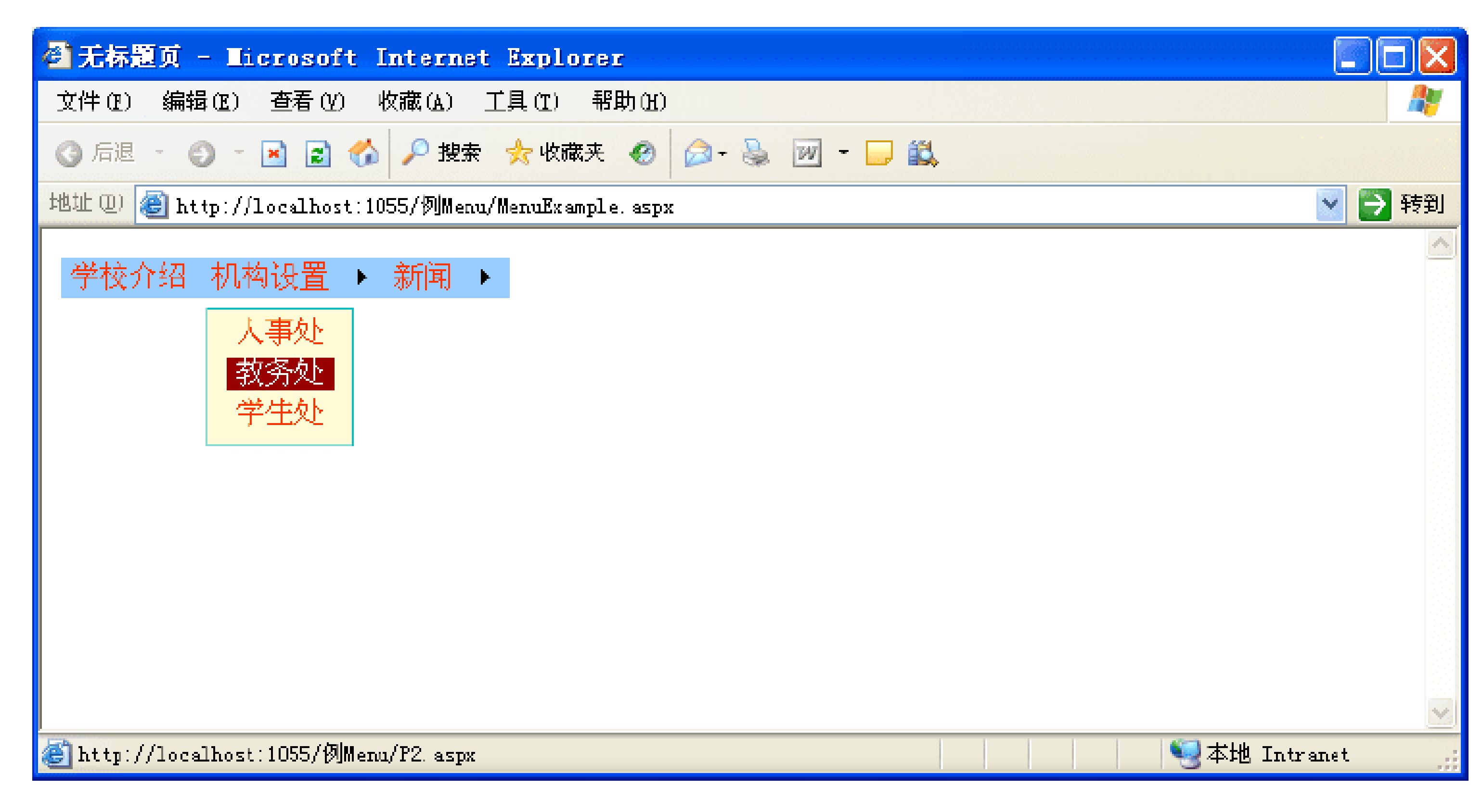


图 5-10　Menu 控件的运行效果

5.2.4　利用 TreeView 控件实现自定义导航

TreeView 控件与 Menu 控件相似，都提供了导航功能。TreeView 控件与 Menu 控件的区别是它不再像 Menu 控件由菜单项和子菜单组成，而是用一个可折叠树显示网站的各个部分。根节点下可以包含多个子节点，子节点下又可以包含子节点，最下层是叶节点。访问者可以快速看到网站的所有部分及位于网站结构层次中的位置。树中的每个节点都显示为一个超链接，被单击时把用户引导到相应的部分。

TreeView 控件也包含很多属性，其中常用属性如表 5-3 所示。

表 5-3　TreeView 控件的常用属性

属　性　名	说　　明
CollapseImageUrl	节点折叠后显示的图像。默认情况下，常用带方框的 + 号作为可展开指示图像
ExpandImageUrl	节点展开后显示的图像。默认情况下，常用带方框的 - 号作为可折叠指示图像
EnableClientScript	是否可以在客户端处理节点的展开和折叠事件。默认值为 true
ExpandDepth	第一次显示 TreeView 控件时，树的展开层次数。默认值为 FullyExpand(即-1)，表示全部展开所有节点
Nodes	设置 TreeView 控件的各级节点及其属性
ShowExpandCollapse	是否显示折叠、展开图像。默认值为 true
ShowLines	是否显示连接子节点和父节点之间的连线。默认值为 false
ShowCheckBoxes	指示在哪些类型节点的文本前显示复选框。共有 5 个属性值：None(所有节点均不显示)、Root(仅在根节点前显示)、Parent(仅在父节点前显示)、Leaf(仅在叶节点前显示)和 All(所有节点前均显示)。默认值为 None

除了表 5-3 所示的 TreeView 的常用属性外，TreeView 控件还有很多与外观相关的属性，

可以用来定制 TreeView 的外观。TreeView 控件的外观属性如表 5-4 所示。

表 5-4　TreeView 控件的外观属性

属　性　名	说　　明
HoverNodeStyle	当鼠标悬停于节点上时，节点的样式
LeafNodeStyle	叶节点的样式
LevelStyle	特殊深度节点的样式
NodeStyle	所有节点的默认样式
ParentNodeStyle	父节点的样式
RootNodeStyle	根节点的样式
SelectedNodeStyle	选定节点的样式

下面通过一个例子演示如何利用 Menu 控件实现自定义导航。

【例 5-8】利用 TreeView 控件实现如图 5-11 所示的导航功能，当单击“节点”时，导航到对应的网页。

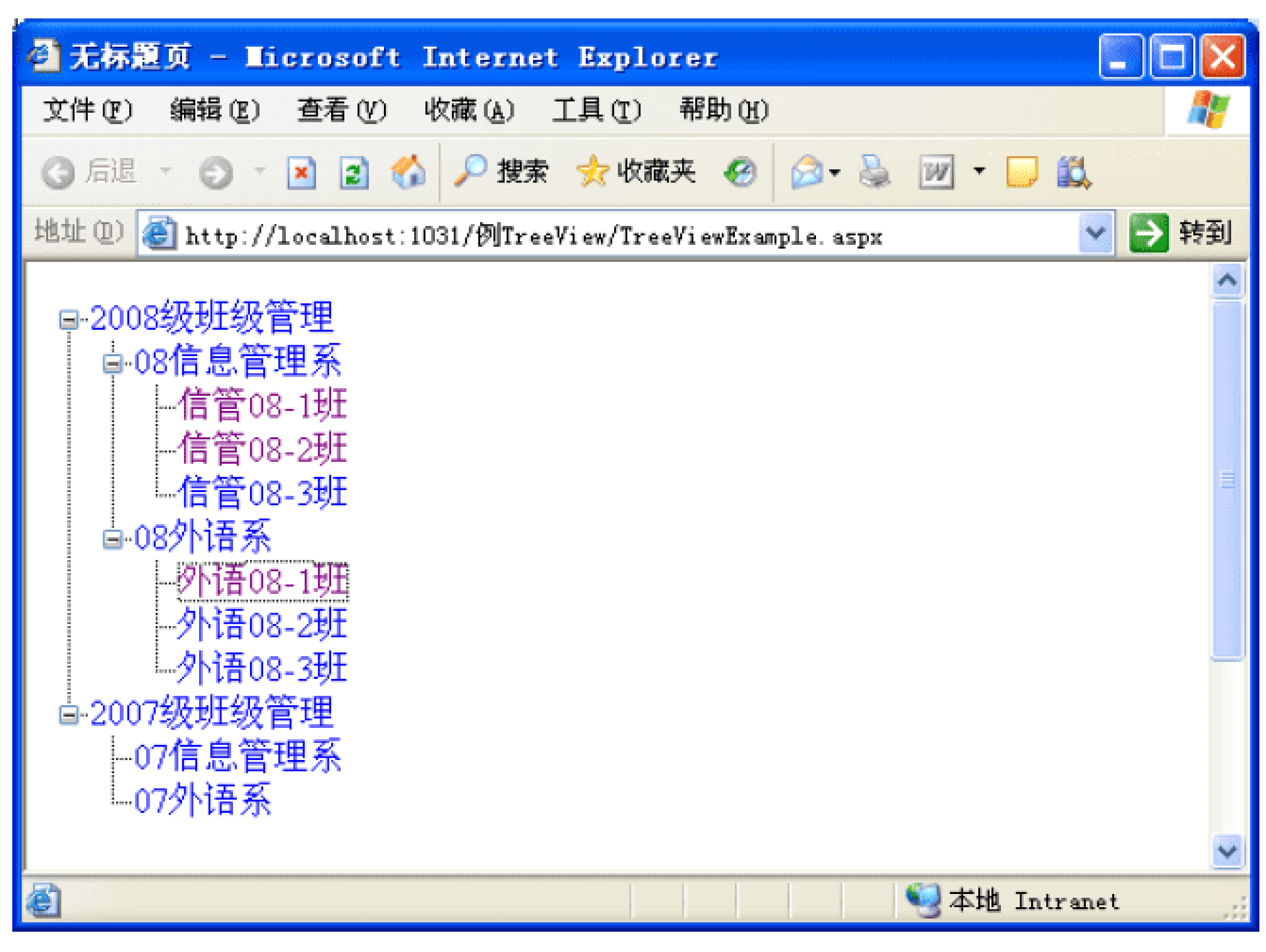


图 5-11　TreeView 导航示例

(1) 运行 VWD 2008，新建一个名为 TreeView_Example 的 ASP.NET Web 应用程序。

(2) 在应用程序中分别添加例子中需要的网页 InformationManage_class1.aspx、InformationManage_class2.aspx、InformationManage_class3.aspx、ForeignLanguage_class1.aspx、ForeignLanguage_class2.aspx 和 ForeignLanguage_class3.aspx。

(3) 在应用程序中添加一个名为 TreeViewExample.aspx 的网页，然后切换到【设计】视图，向页面中拖放一个 TreeView 控件。

(4) 将 TreeView 控件的样式设置为如图 5-12 左侧的样式。

(5) 单击 TreeView 控件右上方的小三角符号，选择【编辑节点】，在弹出的【Treeview 节点编辑器】对话框中，输入各节点名称，如图 5-12 所示。

说明：

为了让读者能看到添加节点后的效果，图 5-12 采用的是添加后重新进入编辑状态看到的效果。如果是第一次添加节点，不会看到图中左侧 TreeView 控件显示的效果。

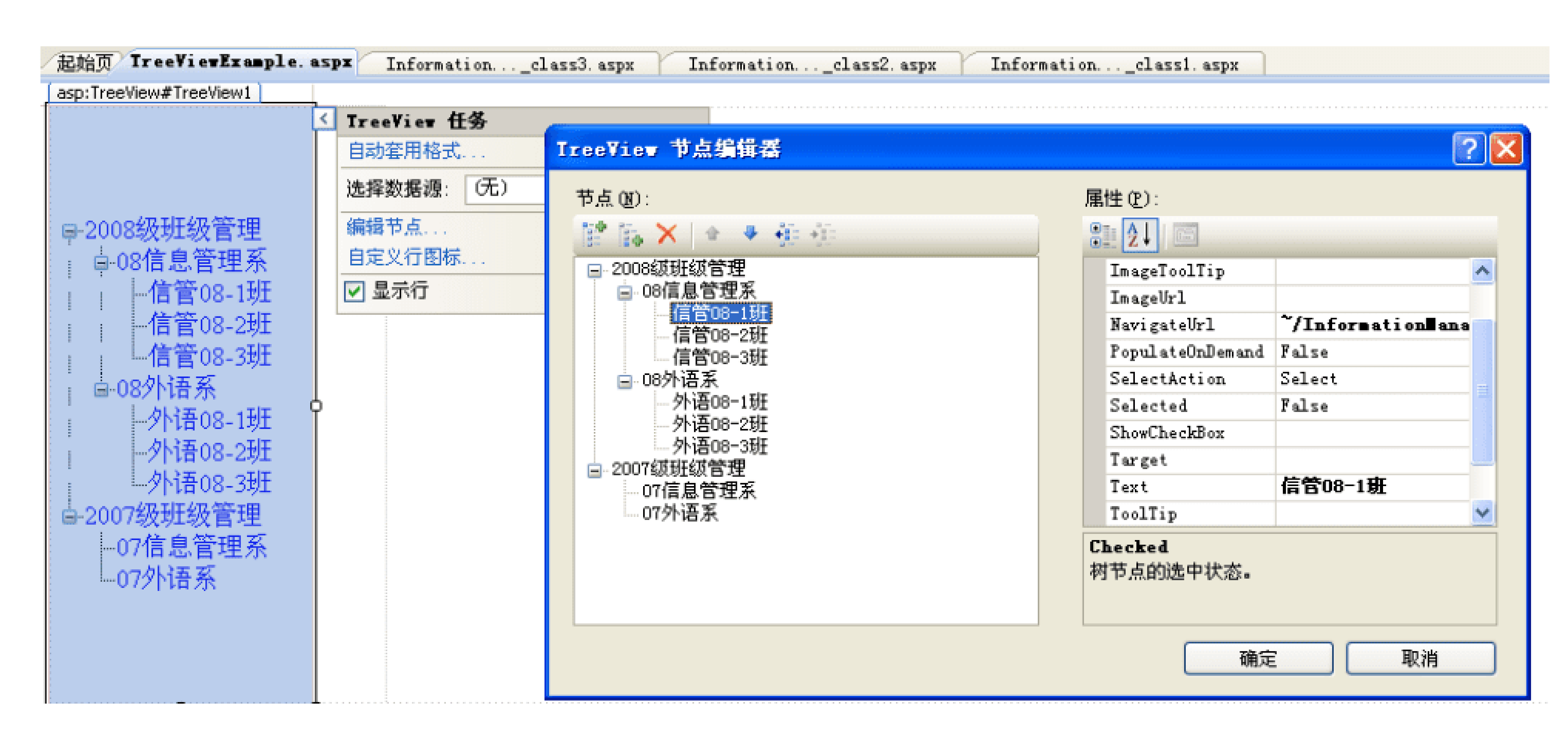


图 5-12　编辑 Treeview 节点

(6) 在【TreeView 节点编辑器】对话框右侧的属性选项中，利用 NavigateUrl 属性设置各节点链接的网页，全部设置完成后，单击【确定】按钮。

(7) 切换到 TreeViewExample.aspx 的【源】视图，将<body>和<body/>之间的部分改为如下内容：

```
<asp:TreeView ID="TreeView1" runat="server"   Target ="_blank"   Height ="376px"
            Width ="165px" ShowLines="True">
            <Nodes>
               <asp:TreeNode Text="2008 级班级管理" Value="2008 级班级管理">
                  <asp:TreeNode Text="08 信息管理系" Value="08 信息管理系">
                     <asp:TreeNode Text="信管 08-1 班" Value="信管 08-1 班"
                       NavigateUrl="~/InformationManage_class1.aspx" >
                     </asp:TreeNode>
                     <asp:TreeNode Text="信管 08-2 班" Value="信管 08-2 班"
                       NavigateUrl="~/InformationManage_class2.aspx" >
                     </asp:TreeNode>
                     <asp:TreeNode Text="信管 08-3 班" Value="信管 08-3 班"
                       NavigateUrl="~/InformationManage_class3.aspx">
                     </asp:TreeNode>
                  </asp:TreeNode>
                  <asp:TreeNode Text="08 外语系" Value="08 外语系">
                     <asp:TreeNode Text="外语 08-1 班" Value="外语 08-1 班"
                      NavigateUrl="~/ForeignLanguage_class1.aspx">
                     </asp:TreeNode>
                     <asp:TreeNode Text="外语 08-2 班" Value="外语 08-2 班"
                      NavigateUrl="~/ForeignLanguage_class2.aspx">
```

```
                    </asp:TreeNode>
                    <asp:TreeNode Text="外语 08-3 班" Value="外语 08-3 班"
                      NavigateUrl="~/ForeignLanguage_class3.aspx">
                    </asp:TreeNode>
                </asp:TreeNode>
            </asp:TreeNode>
            <asp:TreeNode Text="2007 级班级管理" Value="2007 级班级管理">
                <asp:TreeNode Text="07 信息管理系" Value="07 信息管理系">
                </asp:TreeNode>
                <asp:TreeNode Text="07 外语系" Value="07 外语系">
                </asp:TreeNode>
            </asp:TreeNode>
        </Nodes>
</asp:TreeView>
```

(8) 为了便于区别此例中的各个网页，分别在 InformationManage_class1.aspx、InformationManage_class2.aspx、InformationManage_class3.aspx 网页中添加文字“欢迎来到信息管理系 08-1 班”、“欢迎来到信息管理系 08-2 班”和“欢迎来到信息管理系 08-3 班”。接着用同样的方法分别在 ForeignLanguage_class1.aspx、ForeignLanguage_class2.aspx、ForeignLanguage_class3.aspx 网页中添加文字“欢迎来到外语系系 08-1 班”、“欢迎来到外语系系 08-2 班”和“欢迎来到外语系系 08-3 班”。

(9) 切换到 TreeViewExample.aspx 网页，按 F5 键调试运行，分别展开和折叠相应节点，并单击“2008 级班级管理”下“08 信息管理系”和“08 外语系”中的叶节点，观察链接效果。

实验 5　页面切换与导航

1. 实验目的

熟悉页面切换和网站导航的常用方法，掌握利用超链接、按钮的 PostBackUrl 属性、Response.Redirect 和 Server.Transfer 属性实现从一个页面切换到另一个页面的方法，利用 SiteMapPath、Menu 和 TreeView 控件实现网站导航。

2. 实验内容和要求

(1) 运行 VWD 2008，新建一个名为 Pages_Exercise 的 ASP.NET Web 应用程序。

(2) 在 Pages_Exercise 应用程序中，添加一个名为 Home.aspx 的主网页。

(3) 在应用程序中添加 1 个网页 Page1.aspx，利用超链接实现从 Home.aspx 主网页切换到此网页。

(4) 在应用程序中添加 3 个网页 Page2.aspx、Page3.aspx 和 Page4.aspx，分别利用 Button、

LinkButton 和 ImageButton 控件的 PostBackUrl 属性实现从 Page1.aspx 网页切换到此 3 个网页。

(5) 在应用程序中添加 2 个网页 Page5.aspx 和 Page6.aspx，用 QueryString 对象传递数据的方法实现从 Page2.aspx 网页切换到 Page5.aspx 网页，用 Session 对象传递数据的方法实现从 Page3.aspx 网页切换到 Page6.aspx 网页。

(6) 利用站点地图和 SiteMapPath 导航控件实现从 Home.aspx 网页到网页 Page1.aspx、Page2.aspx、Page3.aspx 和 Page4.aspx 的自动网站导航。

(7) 在主网页 Home.aspx 中，利用 Menu 控件实现自定义导航。

(8) 在子网页 Page1.aspx 中，利用 TreeView 控件实现自定义导航。

第6章　页 面 布 局

开发 Web 应用程序通常需要考虑两个方面：功能和外观。其中，外观考虑使 Web 站点更美观一些，包括控件的颜色、图像的使用，页面的布局。ASP.NET 提供了一些可在应用程序中对页面、控件的外观和样式进行自定义的功能，例如可以为某个控件设置字体、背景色和前景色、宽度以及高度等样式，还可以通过合理定位页面中的元素，为用户展现出易于使用且视觉美观的效果。本章将全面来研究 Web 应用程序中样式控制和页面布局所用到的技术和使用方法。

本章的学习目标

- 理解 CSS 的概念，掌握 CSS 的用法
- 理解布局的概念，掌握 CSS 和 Div 布局的方法
- 理解母版页和内容页的概念，掌握创建母版页和内容页的方法
- 理解主题的概念，掌握主题的创建和引用

6.1　CSS 样式控制

样式是指 XHTML 标记元素在浏览器中呈现的形式，如字体的大小、颜色等。在 XHTML 中，最容易、最快速的方法是通过设置标记元素的 style 属性来控制元素的样式，其一般形式为：

```
<元素名称  style="属性名 1:属性值 1;  属性名 2:属性值 2; ……" >显示内容</元素名称>
```

属性名与属性值之间用冒号(:)分隔，如果一个样式中包含多个属性，各属性之间用分号(;)隔开。例如：

```
<div style="font-weight:bold;color:Red;border:solid 2pt navy">这是一个有样式的层</div>
```

在 Internet Explorer 中显示的外观如图 6-1 所示。



图 6-1　包含 style 样式的页面运行效果

这种直接通过设置网页中标记元素的 style 属性来控制的样式称为内联式样式，它的优点就是直观、方便，但是缺点也很明显，假如有多个元素具有相同的样式，那么必须进行多次重复设置，而且修改样式非常繁琐，需要逐一对每个元素进行修改。因此，内联式样式不适合控制具有大量重复样式的元素，而使用 CSS 就可以很好地解决这个问题。本节将介绍 CSS 的概念，如何利用 VWD 2008 定义 CSS，以及 CSS 的样式规则和用法。

6.1.1　CSS 的概念

CSS(Cascading Style Sheet)，中文译为层叠样式表，是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。使用 CSS 样式可以非常灵活并更好地控制网页外观，大大减轻实现精确布局定位、维护特定字体和样式的工作量。

CSS 规定了两种定义样式的方法，分别是内联式和级联式。

1. 内联式样式

直接将样式控制放在单个 HTML 元素内，称为内联式或行内样式。该样式通过 style 属性控制着每个元素的外观，直观但是很繁琐。除非具有相同样式的元素较少，否则很少采用。下面的代码采用内联式来控制各个元素的样式。

【例 6-1】使用 style 样式化一个简单页面。

代码如下：

```
<body style="text-align:center">
  <form id="form1" runat="server">
    <div style="text-align:center; width:400px; border:solid 1px blue">
      <h1 style="font-size:x-large; color:red ">欢迎光临</h1>
      <h2 style="font-size:large; color:blue ">这是一个被 style 修饰的页面</h2>
    </div>
  </form>
</body>
```

2. 级联式样式

在网页的 head 部分定义或导入的样式，称为级联式样式。该样式可以实现将网页结构和表现分离，这样，当修改某些元素的样式时，只需要修改 head 部分定义或引入的样式，该网页内所有具有相同样式的元素都会自动应用新的样式。

级联式样式又可以分为两种方式来控制样式

(1) 在 head 部分直接实现的 CSS 样式，称为内嵌式。这种 CSS 一般位于 HTML 文件的头部，即<head>与</head>标签内，并且以<style>开始，以</style>结束。例如将【例 6-1】代码中的样式抽取出来，经过修改得到如下代码：

```
<head>
 <title>内嵌式样式</title>
 <style Type="text/css">
   <!--
   body{ text-align:center }
   div{ text-align:center; width:400px; border:solid 1px blue }
   h1{ font-size:x-large；color:red}
   h2{ font-size:large;   color:blue }
  -->
 </style>
</head>
<body>
 <form id="form1" runat="server">
  <div>
   <h1>欢迎光临</h1>
   <h2>这是一个被 style 修饰的页面</h2>
  </div>
 </form>
</body>
```

其中<style>与</style>直接是样式的内容，在{ }前面可以写样式的类型和名称。{ }中是样式的属性。这种方法是经常被使用的添加样式表的方法。

可见，采用内嵌式比内联式方便了很多，body 内的代码也相对简洁，修改某个元素的样式时只需要修改 head 内的代码即可。但是，内嵌式只解决了一个网页内部结构和表现分离的问题，一般情况下网站是由很多网页组成，如果不同网页中的某些元素采用了相同的样式，仍然需要分别设置，因此，将样式放在一个单独的 CSS 文件中，然后通过为每个网页引入该文件来实现统一的外观成为一种更好的选择。

(2) 在 head 部分通过导入以扩展名为.css 的文件来实现 CSS 样式，称为链接式。利用这种方法在网页中可以调用已经定义好的样式表来实现样式表的应用，定义好的样式表通常单独以文件的形式存放在站点目录中。这种方法实现了将网页结构和表现的彻底分离，最适合大型网站的 CSS 样式定义。

例如将【例 6-1】代码中的样式抽取出来以文件的形式存放，经过修改得到如下代码：

```
<head>
    <title>链接式样式</title>
    <link href="Style.css" rel="stylesheet" type="text/css" />
</head>
<body>
```

```
        <form id="form1" runat="server">
            <div>
                <h1>欢迎光临</h1>
                <h2>这是一个被 style 修饰的页面</h2>
            </div>
        </form>
    </body>
```

刚才的样式内容都被保存到一个名为 style.css 的文件中，文件内容如下所示：

```
body
{
    text-align:center;
}
div
{
    text-align:center;
    width:400px;
    border:solid 1px blue;
}
h1
{
    font-size:x-large;
    color:red;
}
h2
{
    font-size:large;
    color:blue;
}
```

在引用样式的标记<link>中，ref 属性规定了 XHTML 与被链接文件的关系，href 属性指定了要链接的样式表文件的 URL，type 属性则规定了链接文件的类型。

注意：

该文件是和当前页面在同一个目录下存放，如果不在同一个目录下存放，相应的<link href=" "...>标记中 href 属性的值要有所改变。

此外，如果某个元素既引用了链接样式文件中定义的样式，又在 head 部分定义了新的样式，或者在元素内部通过 style 属性定义了新的样式，那么该标记元素最终呈现的效果会是什么样呢？下面通过一个例子来说明这个问题。

【例 6-2】样式嵌套举例。

```
<head>
  <title>链接式样式</title>
```

```
    <style Type="text/css">
    <!--
     h1{ font-weight:bold }
     h2{ color: yellow}
    -->
    </style>
    <link rel="stylesheet" href="style.css" type="text/css">
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <h1 style=" font-size:small ">欢迎光临</h1>
      <h2 style=" font-weight:bold ">这是一个被 style 修饰的页面</h2>
    </div>
  </form>
</body>
```

其中，style.css 文件的内容同上。运行这个 XHMTL 文件，在浏览器中可以看到，h1 元素内的文字以粗体、小号、红色显示，而 h2 元素内的文字则以粗体、大号、蓝色显示。

可见链接式样式中 h1 元素的 font-size 属性和内嵌式样式中 h2 元素的 color 属性都没有起作用，而不冲突的样式则都会起作用。这就是样式嵌套中的冲突问题，浏览器解决这种问题的方法就是一旦发现样式冲突，则通过“就近使用”原则，采用距离该元素最近的样式进行显示，而不冲突的样式则通过顺序组合后形成最终样式进行显示。

设计者可以根据实际情况选择一种或多种样式控制方法进行样式定义。一般情况下，在样式表(.css)文件中定义适合大多数网页公用的样式，在网页内部采用内嵌式定义该页面特有的样式，内联式样式定义个别元素的样式，再结合可视化的开发工具，从而使样式控制真正灵活、方便。

在 VWD 2008 中，设置样式的方法有两种，一种是在【源】视图下直接设置样式，另一种是利用可视化界面设置样式。

6.1.2　在源视图下设置样式

创建一个新的网页，系统自动打开页面【源】视图，利用系统提供的智能提示功能，可以方便地设置各种元素的样式，下面通过具体步骤说明样式设置的方法。

(1) 打开 VWD 2008，新建一个名为 StyleTest 的 ASP.NET Web 应用程序，在 Default.aspx 源文件的 body 部分输入【例 6-1】的代码，输入时可以看到 VWD 2008 所提供的智能提示工具，如图 6-2 所示。

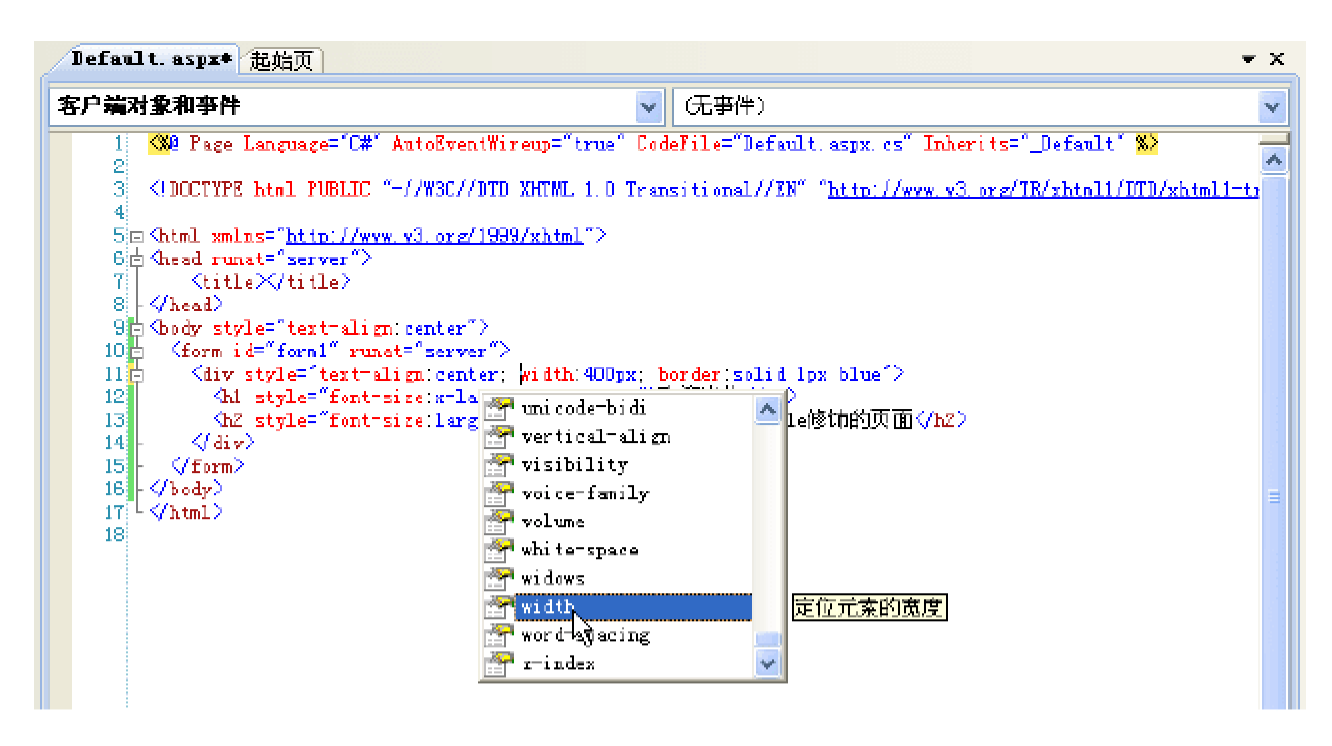


图 6-2　源视图中提供的智能代码提示

(2) 完成代码输入后切换到【设计】视图，可以查看页面的显示效果，如图 6-3 所示。

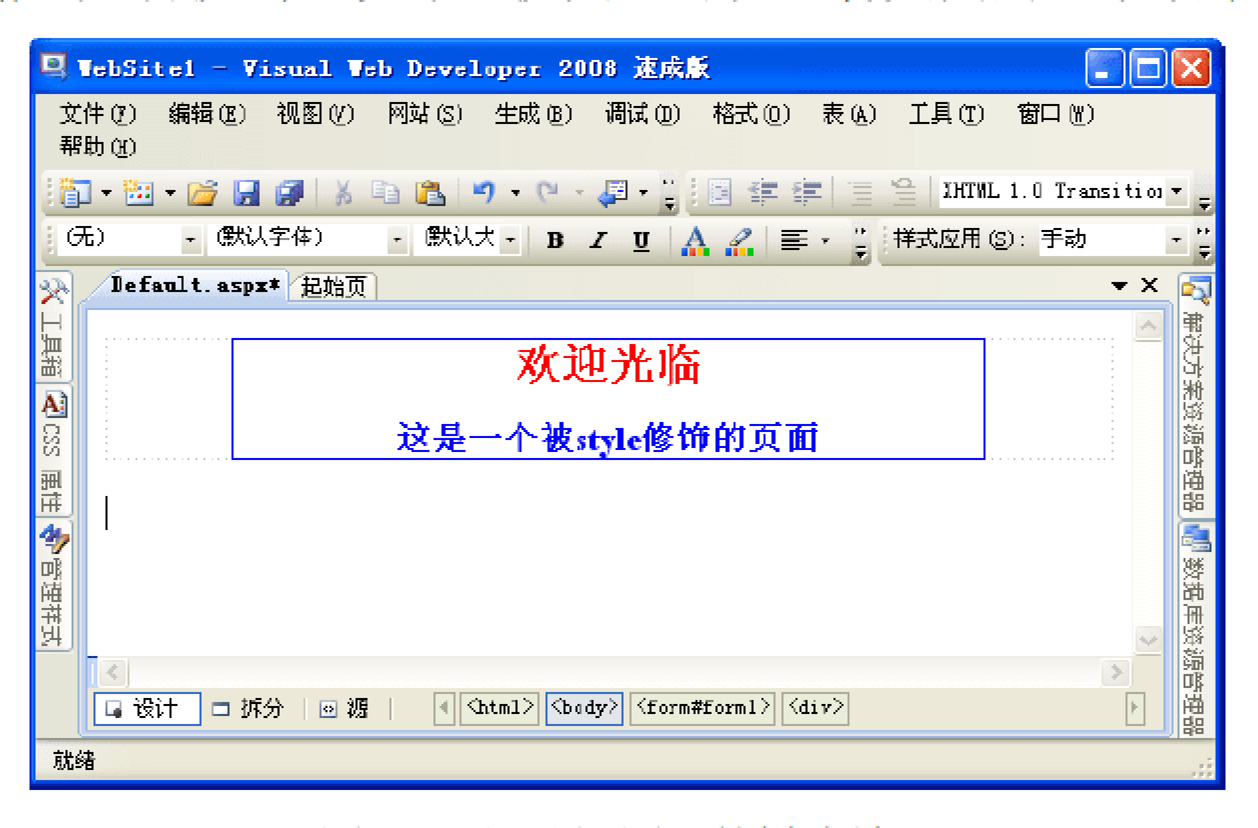


图 6-3　设计视图下的样式效果

虽然在【源】视图下输入样式信息时，系统将提供智能提示以帮助用户完成样式输入，但是初学者往往不了解到底都有哪些样式可供使用，对于提供的选项也不知道什么含义，因此，利用 VWD 2008 的可视化界面可以更为方便地完成样式的定义。

6.1.3　在可视化窗口中设置样式

利用可视化窗口设置样式的方法有很多，可以在【源】视图或者【设计】视图下选中某个标记元素，然后在属性面板中 style 属性值的某个地方单击，将在样式信息后看到省略号按钮(…)，如图 6-4 所示。单击该按钮将看到【修改样式】对话框，如图 6-5 所示。

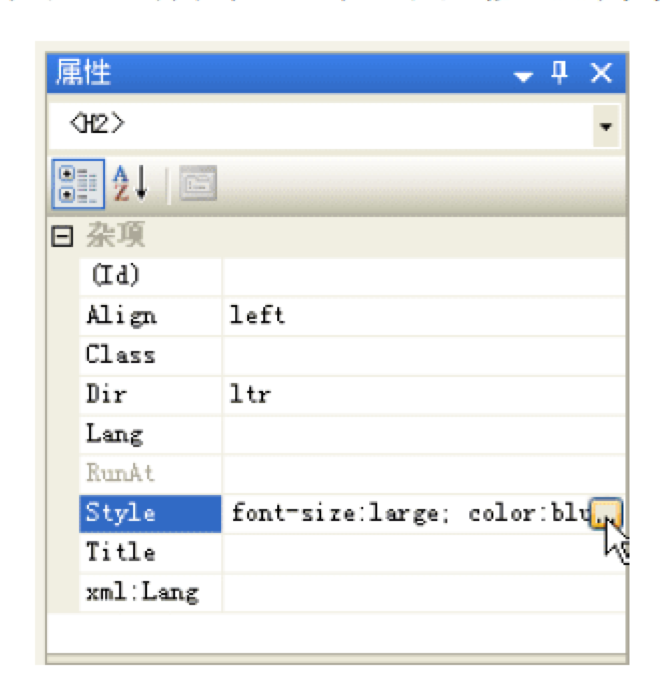


图 6-4　属性面板的 style 属性

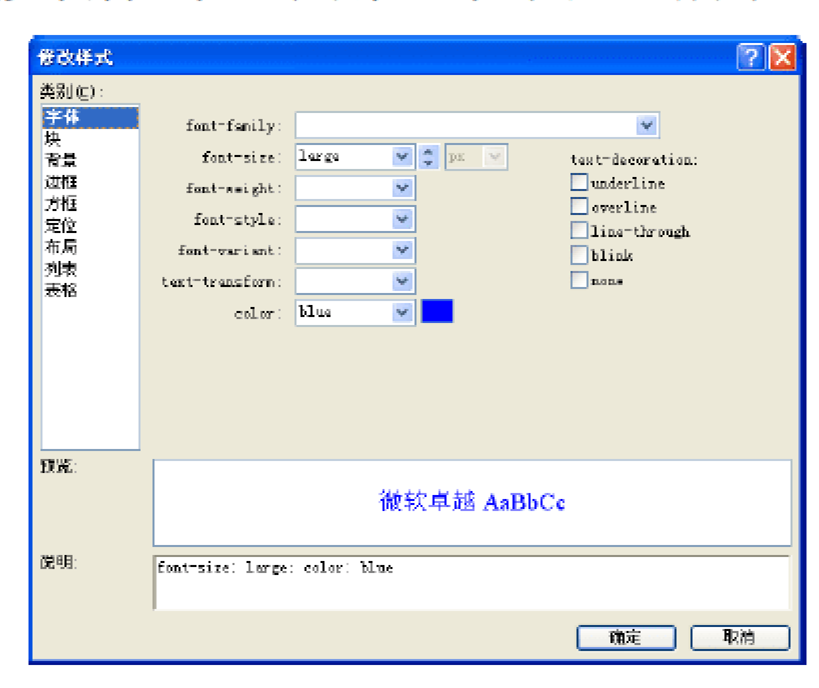


图 6-5　【修改样式】对话框

该对话框分为两个窗格，左窗格列出 9 个类别，当选择某个类别时，右窗格显示所选类别下的选项。选择了样式选项并单击【确定】按钮后，新的样式定义将自动在【源】视图中生成，还可以在【设计】视图下查看最新的效果。

这种方法只能将定义的样式属性以内联式生成，放在每个元素的 style 属性中，内联式样式的缺点前面已经讲过。要想定义内嵌式样式也非常容易，具体步骤如下。

(1) 切换到 【设计】视图中，在【样式应用】工具栏的【目标规则】列表中，单击【应用新样式】，如图 6-6 所示。

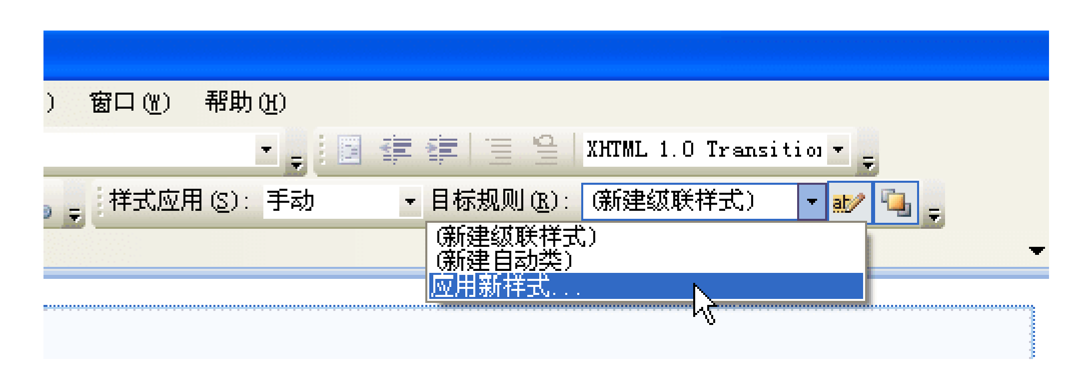


图 6-6　新建样式

(2) 此时将显示【新建样式】对话框，该对话框和如图 6-5 所示的【修改样式】对话框相似，不同的是【新建样式】对话框中包含了选择器用于选择对哪一个标记进行定义，以及通过定义位置将当前定义存放到哪里。如图 6-7 所示。

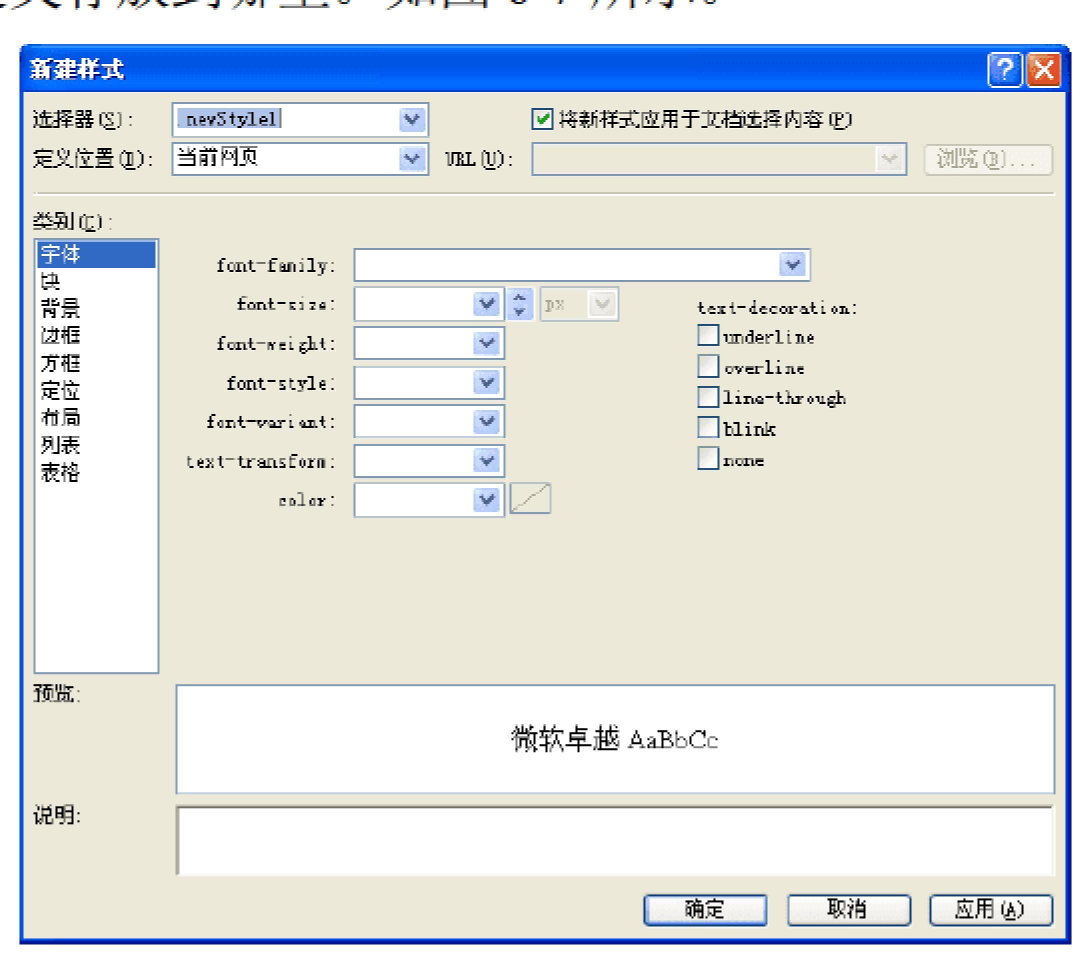


图 6-7　【新建样式】对话框

在【选择器】列表中单击 h1，就可以创建应用于所有 h1 元素的样式。

注意，【定义范围】列表设置为“当前页面”，这指示该样式规则在当前页的 style 元素中创建。若想查看已创建的样式规则，可以切换到【源】视图并滚动到 style 元素，该元素位于 head 元素内。

也可以通过选择菜单【格式】中的【新建样式】命令来定义内嵌式样式。

想要对正在定义的内嵌式样式规则进行修改也非常简单，具体步骤如下。

(1) 单击某个想要修改样式的元素，如文本“欢迎光临”。此时，选中部分以蓝色框包围并有一个标签指示 h1 元素已选中。

(2) 选择【视图】菜单中的【CSS 属性】命令，打开【CSS 属性】面板，可以看到 h1 元素的 CSS 属性列表，如图 6-8 所示。

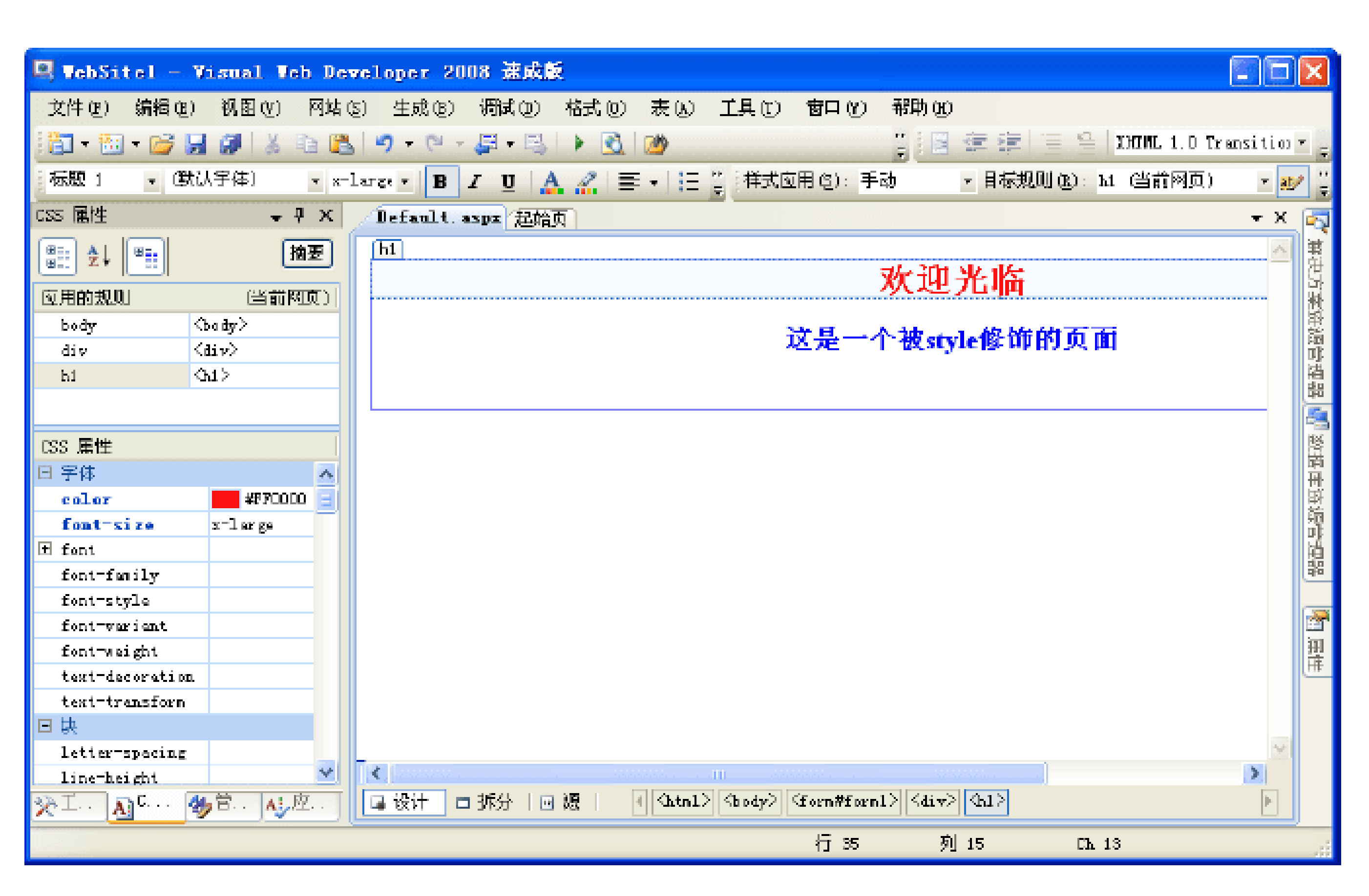


图 6-8　修改 CSS 属性

在 CSS 属性列表中所做的修改可以立即通过【设计】视图显示出来，如果想要观察具体的样式代码，可以通过切换到【源】视图并滚动到 style 元素来进行查看。

使用 CSS 的另一个有效方法是将样式规则放入样式表中。然后，所有页面都可以引用这些样式，这样可以使这些页面看起来非常一致。创建样式表使用的技术与创建新页面所用的相同。具体步骤如下。

(1) 在【解决方案资源管理器】中，右击网站的名称，然后单击【添加新项】。在【Visual Studio 已安装的模板】下选择【样式表】。

(2) 在【名称】文本框中，输入 NewStyle.css，然后单击【添加】按钮。编辑器将打开，其中显示一个包含空 body 样式规则的新样式表。

(3) 打开或切换到 Default.aspx 页，然后切换到【设计】视图。在【格式】菜单中单击【管理样式】命令，打开【管理样式】面板。单击【附加样式表】，此时将显示【选择样式表】对话框，选择 NewStyle.css 文件，然后单击【确定】按钮，名为 NewStyle.css 的新选项卡在【管理样式】窗口中创建。如图 6-9 所示。

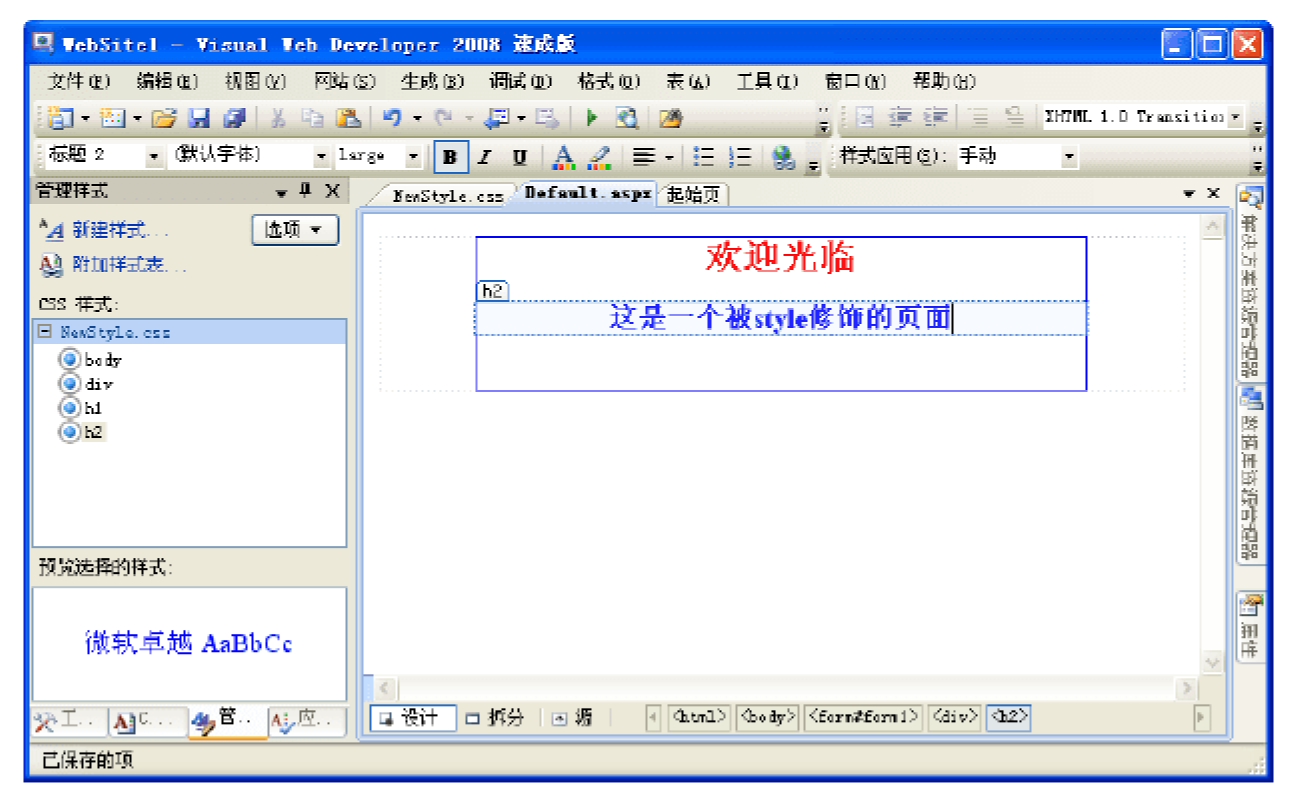


图 6-9　新建 CSS 样式表文件

可以通过多种方式为页面指定样式表。最简单的方法是在【源】视图中将文件从【解决方案资源管理器】拖到页面的头元素中或者直接将文件拖到【设计】视图中。

6.1.4　样式规则

一个样式表由若干个样式规则组成。样式规则是指网页元素的样式定义，包括元素的显示方式以及元素在页中的位置等。在【解决方案资源管理器】中，添加一个样式表文件 StyleSheet1.css，在大括号的外边单击鼠标右键，然后在快捷菜单中选择【添加样式规则】命令，如图 6-10 所示，即弹出如图 6-11 所示的【添加样式规则】对话框。

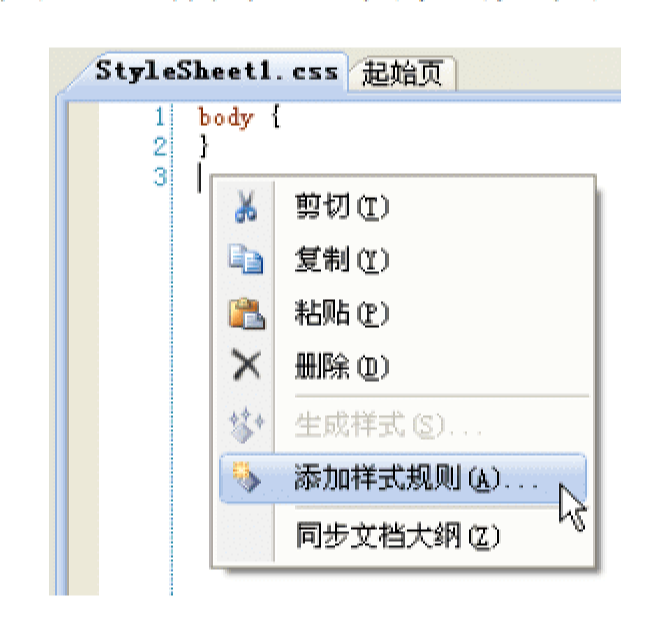




图 6-10　选择【添加样式规则】命令

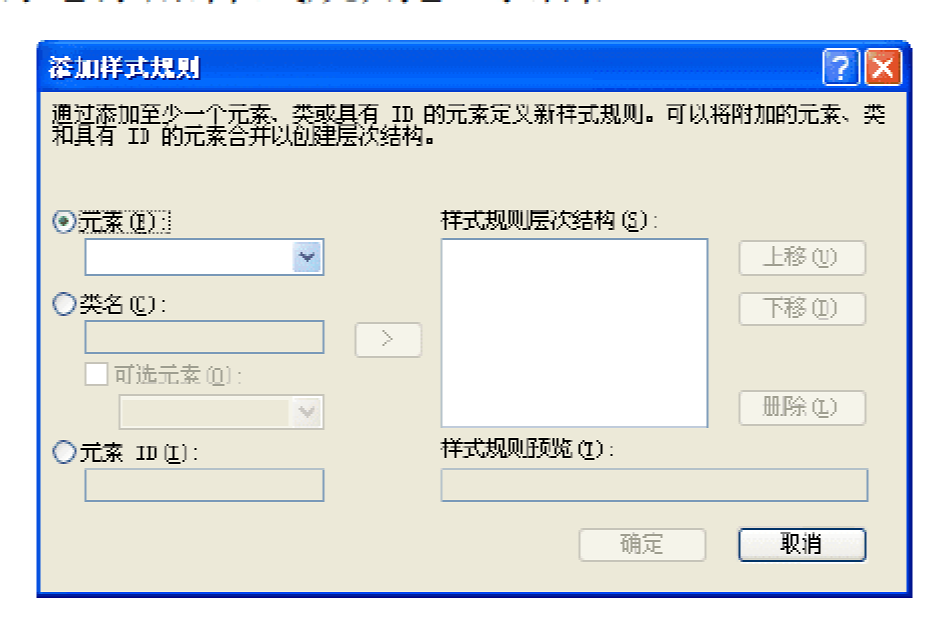


图 6-11　【添加样式规则】对话框

在【添加样式规则】对话框中选择某个元素，或者定义一个类，或者定义一个元素 ID，单击【确定】按钮即可添加一个新的样式规则。例如添加一个元素 h1，在样式表文件中可以看到新建的样式规则。

```
h1
{
}
```

该规则默认是仅有元素名称的空规则，在大括号内单击鼠标右键，在快捷菜单中选择【生成样式】命令，即弹出【修改样式】对话框。

可以看到，无论是定义内嵌式样式还是链接式样式，每个样式的定义格式都是一样的：

```
样式定义选择符{ 属性 1:值 1; 属性 2:值 2;  ……  }
```

其中，样式定义选择符是指样式定义的对象，可以是 HTML 标记元素，还可以是用户自定义的类、用户自定义的 ID、伪类和伪元素等。

1. 标记选择符

任何 HTML 元素都可以是一个 CSS 的标记选择符。标记选择符仅仅是指向特别样式的元素。例如下面规则中的标记选择符是 div。

```
div{   text-align:center }
```

该样式规则的含义是，div 块内的所有内容都居中显示。图 6-11 中的【元素】下拉列表中提供了所有可供使用的标记选择符。

2. 类选择符

每一个标记选择符都能自定义不同的类，从而允许同一元素具有不同的样式。指定某个

标记选择符内的自定义类的一般形式为：

标记选择符. 类名{样式属性 1:值 1; 样式属性 2:值 2; ……}

例如：

```
<style type="text/css">
  p.one{
    color:red;
  }
  p.two{
    color:blue;
  }
</style>
```

在代码中引用类选择符的方法是通过元素的 class 属性来实现的。

```
<p class="one">类别选择器 1</p>
<p class="two">类别选择器 2</p>
```

其含义是在 p 中引用 one 会以红色样式显示，在 p 中引用 two 会以蓝色样式显示。

在图 6-11 中，先选择【类名】单选按钮，在文本框中输入 one，然后选中【可选元素】复选框，在其下拉列表中选择 p 元素，即可自动生成 p.one 样式规则。如图 6-12 所示。

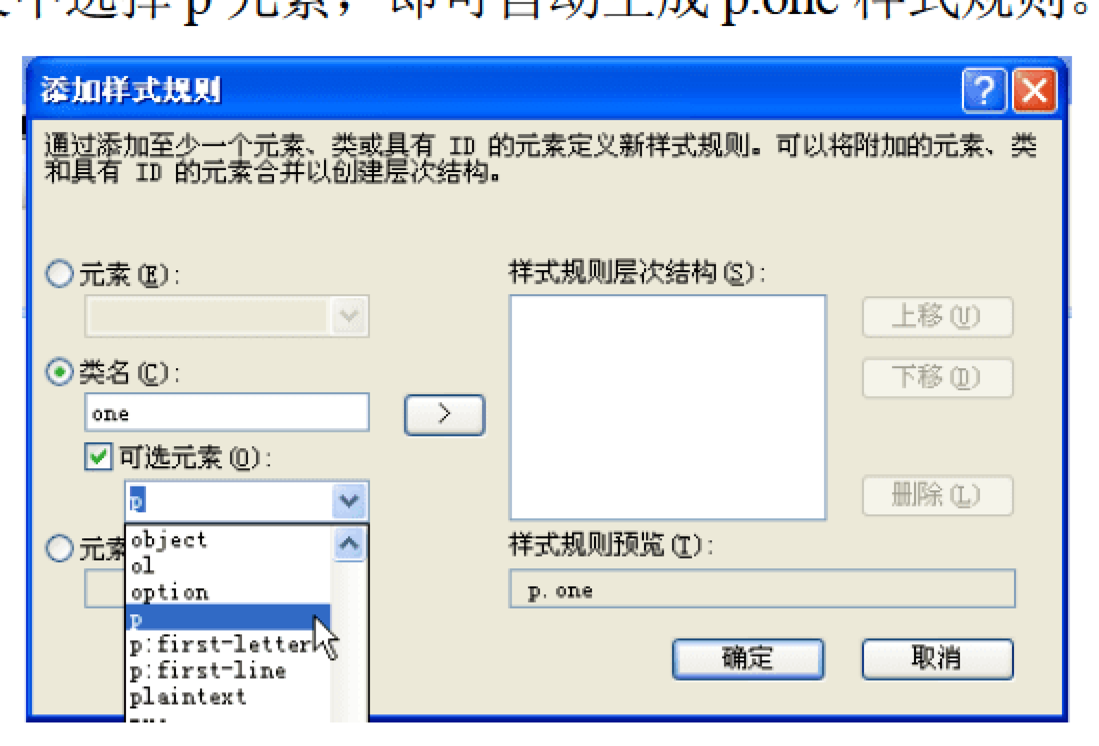


图 6-12　添加 p.one 样式规则

类选择符的定义也可以与标记选择符无关，这样，类选择符可以应用于任何元素。这种自定义类选择符的形式如下：

. 类名{样式属性 1:值 1; 样式属性 2:值 2; ……}

例如：

```
<style type="text/css">
  .note{
    color:red;
  }
</style>
```

```
<h1 class="note">类别选择器 1</p>
<h2 class="note">类别选择器 2</p>
```

在这个例子，note 类选择符可以被用于任何元素。

3. ID 选择符

ID 选择符用于分别定义每个具体元素的样式。一个 ID 选择符的指定要有指示符#在名字前面。使用时通过指定元素的 id 属性来关联。例如：

```
#index { color:blue }
```

引用时，使用 id 属性声明即可。

```
<p id="index">本段落的颜色为蓝色</p>
```

可以看出，自定义 ID 选择符与自定义类选择符的定义方式非常相似，但是两者在使用上是有区别的。在同一个网页中，多个标记元素可以使用同一个自定义类选择符，而 ID 选择符只能为某一个标记元素使用。这种选择符应该尽量少用，因为它具有一定的局限。但是如果在一个元素的样式定义中，既有标记选择符，又有自定义类选择符和自定义 ID 选择符，那么自定义 ID 选择符的优先级最高，其次是自定义类，标记选择符的优先级最低。

4. 伪类

伪类是 CSS 中非常特殊的类，能自动地被支持 CSS 的浏览器所识别。伪类可以指定 XHTML 中的 A 元素以不同的方式显示链接(links)、已访问链接(visited links)和可激活链接(active links)。其中，一个已访问链接可以定义为不同颜色的显示，甚至不同字体大小和风格。

CSS 中用 4 个伪类来定义链接的样式，分别是：a:link、a:visited、a:hover 和 a : active，例如：

```
a:link{font-weight : bold ;text-decoration : none ;color : #C00000 ;}
a:visited {font-weight : bold ;text-decoration : none ;color : #C30000 ;}
a:hover {font-weight : bold ;text-decoration : underline ;color : #F60000 ;}
a:active {font-weight : bold ;text-decoration : none ;color : #F90000 ;}
```

以上语句分别定义了“链接、已访问过的链接、鼠标停在上方时、点下鼠标时”的样式。注意，必须按以上顺序写，否则显示可能和预想的不一样。记住它们的顺序是 LVHA。

5. 关联选择符

关联选择符是一个用空格隔开的两个或更多的单一标记选择符组成的字符串。一般格式如下：

```
选择符 1　选择符 2　…… {属性:值; ……}
```

这些选择符具有层次关系，并且它们的优先级比单一的标记选择符大。例如：

```
p h1 { color:red }
```

这种定义方式只对 p 所包含的 h1 元素起作用，单独的 p 或者单独 h1 元素均无法采用该样式。通过【添加样式规则】对话框可以添加关联规则，这种方式不仅适用于标记选择符，还可以关联自定义用户类，自定义 ID 以及任何样式选择符中。如图 6-13 所示，依次添加各种选择符形成关联规则。

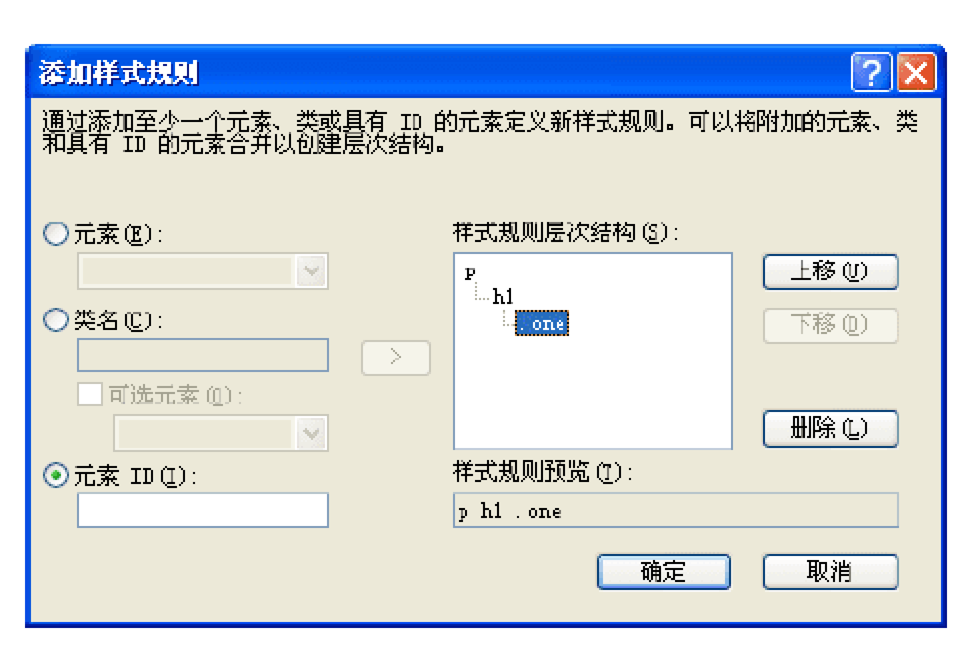


图 6-13　关联选择符的定义

这样一来，在此规则内定义的样式，只能对 p 内包含的 h1 中 class 属性为 one 的内容起作用。

6. 并列选择符

如果有多个不同的元素定义的样式相同，则可以使用并列选择符简化定义。例如：

```
h1,h2,h3{ color:blue}
```

每个元素之间用逗号隔开，表示所有的 h1、h2、h3 标记中的内容都将以蓝色显示。

6.2　页面布局

除了为页面的内容设置样式，页面元素的布局和定位是否合理也是衡量网页设计是否美观的重要指标。这一节将介绍网页的基本布局方式，页面元素的定位，以及表格布局和层布局。

6.2.1　网页的基本布局方式

常见的网页布局方式有左对齐、居中和满宽度显示。默认情况下，网页内容水平左对齐，然而，在实际页面中，经常看到的大部分页面都是水平居中和满宽度显示的。因此，这里仅介绍两种非常常用的布局实现方法。

1. 页面水平居中

实现的方法是在 body 的 style 样式中设置 text-align 属性的值为 center 即可。如果还希望页面的宽度固定，则可以通过设置 div 的 width 属性来实现。演示代码如下：

```
<body style="text-align:center; ">
  <form id="form1" runat="server">
```

```
      <div id="div1" style="width:760px; text-align:center; height:200px"></div>
  </form>
  </body>
```

2. 页面满宽度显示

实现的方法是将 div1 的固定宽度设置为百分比即可，这样宽度就会随显示界面的大小自动调整。演示代码如下：

```
<body style="text-align:center; ">
   <form id="form1" runat="server">
     <div id="div1" style="width:98%; text-align:center; height:200px"></div>
   </form>
</body>
```

这种方式的优点是无论浏览器是否最大化显示，都不会出现横向滚动条，如网易的 126 免费邮箱(www.126.com)。缺点是页面元素相对位置不固定，不利于用户和窗体之间的操作。

6.2.2　页面元素定位

页面元素的定位分为流布局和坐标定位布局两种，其中，坐标定位布局又分为绝对定位和相对定位，这里仅介绍流布局和坐标绝对定位。

1. 流布局 static

如果采用该布局，则页面中的元素将按照从左到右、从上到下的顺序显示，各元素之间不能重叠。如果不设置元素的定位方式，则默认是流式布局。

2. 坐标绝对定位 absolute

在使用坐标绝对定位之前，必须先将 style 元素的 position 属性的值设置为 absolute，然后就可以由 style 元素的 left、top、right、bottom 和 z-index 属性来决定元素在页面中的绝对显示位置。left 属性表示元素的 x 坐标，top 属性表示元素的 y 坐标，坐标的位置是以它最近的具有 position 属性的父容器为参照物的。具体效果参看下面的例子。

【例 6-3】新建一个.aspx 页面，修改页面<body>内的代码为下面的内容。

```
<body>
   <form id="form1" runat="server">
      <div id="div1" style="border: 1px #000080 solid; text-align: left; width: 400px; height: 200px;">
         <div id="div2" style="width: 200px; height: 120px; text-align: left; border: 1px #00FF00 solid;
background-color: #808080">
            <div id="div3" style="position: absolute; top: 70px; left: 130px; width: 150px; height: 100px;
border: 2px #800000 solid; background-color: #FFFF00">
               <div id="div4" style="position: absolute; left: 30px; top: 30px; width: 100px; height: 60px;
border: 3px #FF00FF solid; background-color: #00FFFF">
               </div>
```

```
            </div>
          </div>
        </div>
      </form>
    </body>
```

然后切换到【设计】视图，观察显示的效果，如图 6-14 所示。运行该页面，可以看到，无论浏览器窗口如何变化，各层之间的位置仍然保持不变。

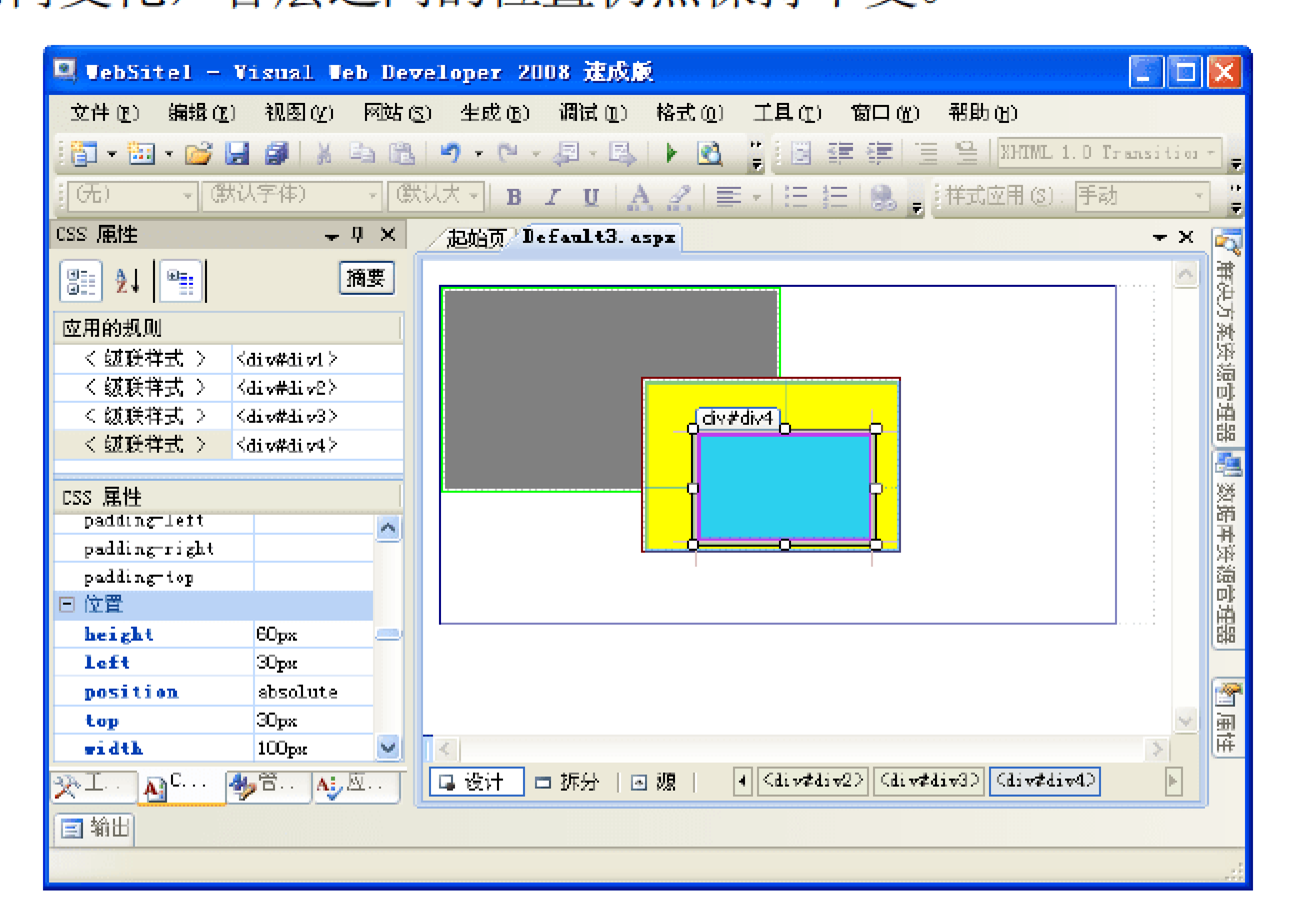


图 6-14　绝对定位的页面效果

具有不相同 z-index 值的元素可以重叠，其效果就像多张透明的纸按顺序叠放在一起。其中，z-index 值大的元素会覆盖 z-index 值小的元素。修改【例 6-3】的代码，为 div3 增加 z-index 属性，如下所示：

```
<div id="div3" style="position: absolute;    ……    z-index: -1;">
```

切换到【设计】视图，观察显示的效果，如图 6-15 所示。

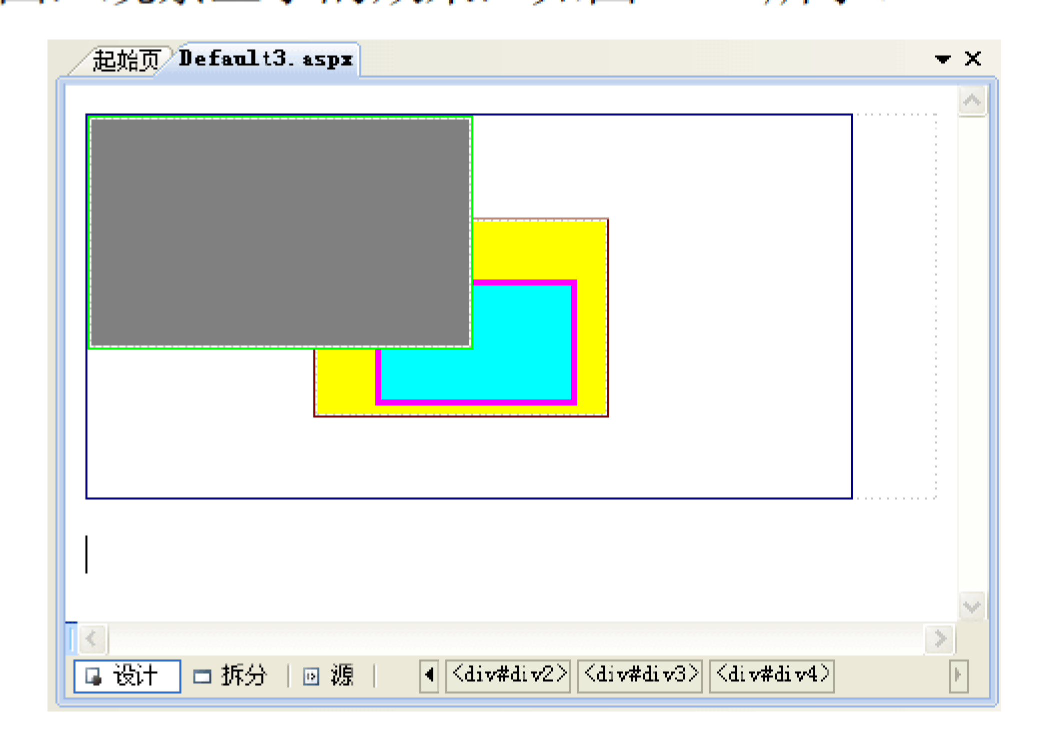


图 6-15　设置 z-index 属性的效果

采用坐标定位的方式可以精确地将元素放在页面中相应的位置显示，但是由于不同浏览器在显示方面存在的差异，也会给整体页面布局带来混乱的效果，解决这个问题的方法就是利用表格来进行布局。

6.2.3　表格布局

利用表格可以将网页中的内容合理地放置在相应的区域，每个区域之间互不干扰。例如，设计一个表格用来布局网页首页，实现的效果如图 6-16 所示。

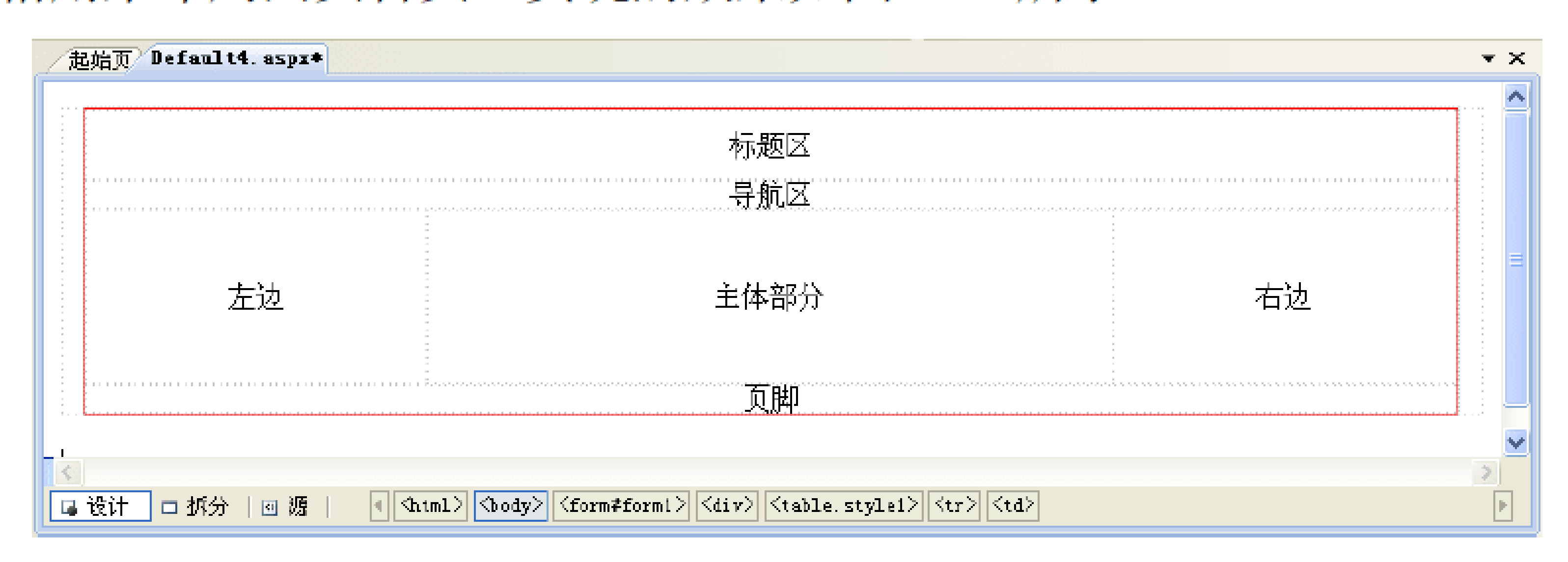


图 6-16　表格布局

从图 6-16 中可以看到，表格中定义了 1 个标题区，1 个导航区，1 个页脚区，中间又分成 3 个区，这就需要先创建一个 4 行 3 列的表格，然后再通过详细设置达到图 6-16 中的效果。该实例的实现步骤如下。

(1) 在【解决方案资源管理器】中，右击网站的名称，然后单击【添加新项】，新建一个.aspx 页面，设置 body 元素的 style 属性为"text-align:center"，div 元素的 style 属性为"width: 780px; text-align:center"。

(2) 切换到【设计】视图，将鼠标光标停在 div 标记内。选择菜单【表】中的【插入表】命令，打开【插入表格】对话框，定义表格大小为 4 行 3 列，指定宽度为 100%，边框值为 1，边框颜色为红色。如图 6-17 所示。

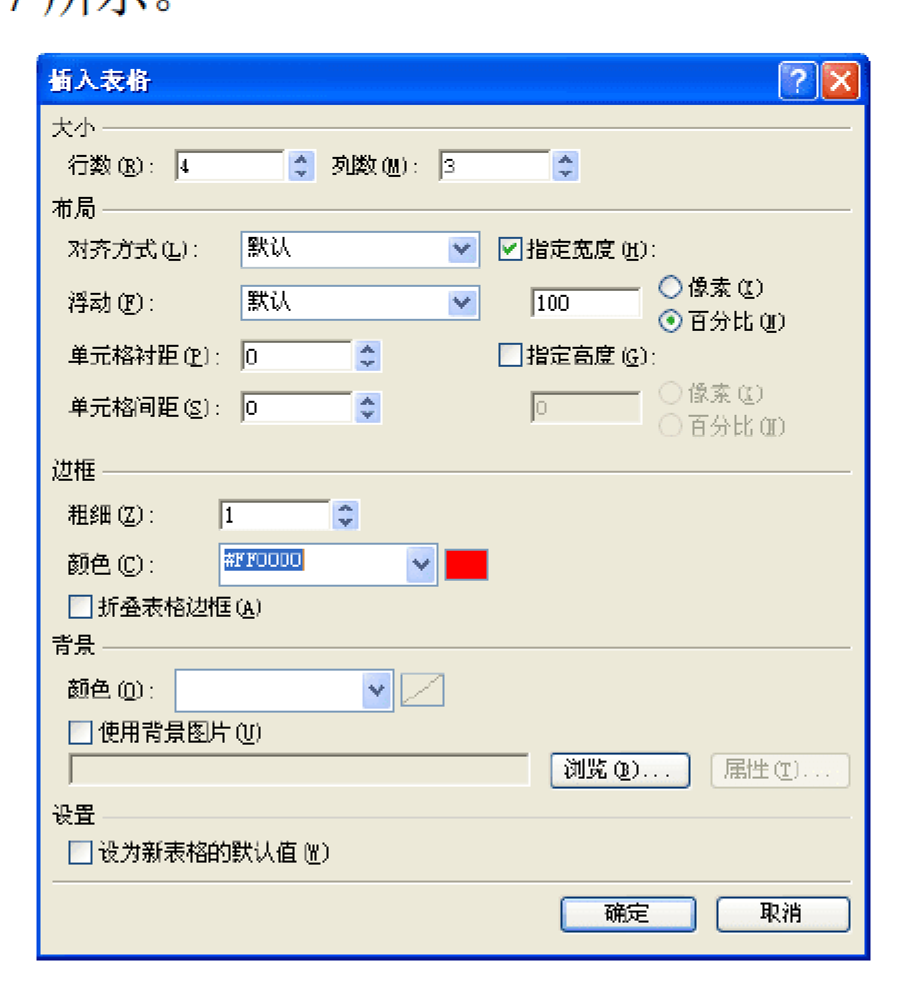


图 6-17　新建表格

经过详细设置后的代码可以通过切换到【源】视图中查看，如【例 6-4】所示。

【例 6-4】表格布局举例。

```
<head runat="server">
    <title></title>
    <style type="text/css">
```

```
        .style1
        {
            width: 100%;
            border: 1px solid #FF0000;
        }
    </style>
</head>
<body style="text-align:center">
    <form id="form1" runat="server">
        <div style="width: 780px; text-align:center">
            <table cellpadding="0" cellspacing="0" class="style1">
                <tr><td colspan="3" height="40px">标题区</td></tr>
                <tr><td colspan="3">导航区</td></tr>
                <tr>
                  <td height="100px" width="200px">左边</td>
                  <td>主体部分</td>
                  <td width="200px">右边</td>
                </tr>
                <tr><td colspan="3">页脚</td></tr>
            </table>
        </div>
    </form>
</body>
```

整个表格的最大列数是 3，所以标题区、导航区和页脚部分的 colspan 属性值为 3，表示对应行占 3 列，中间部分左右两边宽度为 200px，主体部分则没有指定宽度，此时，浏览器会自动根据整个表格的宽度调整该部分的宽度。

表 6-1 中列出了表格中部分常用的属性。

表 6-1　常用表格属性

属 性 名	含　义
Border	表示边框宽度，如果设置为 0，表示无边框，此时默认 frame=void，rules=none；可以设置为大于 0 的值来显示边框，此时默认 frame=border，rules=all
Cellspacing	表示单元格间距(表格和 tr 之间的间隔)
Cellpadding	表示单元格衬距(td 和单元格内容之间的间隔)
Frames	表示如何显示表格边框，void：无边框(默认)；above：仅有顶部边框；below：仅有底部边框；hsides：仅有顶部和底部边框；vsides：仅有左右边框；lhs：仅有左边框；rhs：仅有右边框；box 和 border：包含全部 4 个边框
Rules	表示如何显示表格内的分隔线，all：显示所有分隔线；cols：仅显示列线；rows：仅显示行线；groups：仅显示组与组之间的分隔线

表格布局的最大优点就是简单直观。但是如果将整个网页的元素都包含在表格内，则浏览器会将整个表格全部下载完毕后才显示表格中的内容，因此网页显示速度慢。此外，表格布局也不利于网页结构和表现的分离。解决该问题的方法就是网页整体采用 DIV 和 CSS 进行层布局，局部用表格进行布局。这是当前 Web 标准推荐的最佳布局方法。

6.2.4　DIV 和 CSS 布局

层布局最核心的标签就是 DIV。DIV 是一个容器，在使用时以<DIV></DIV>形式存在。在 XHTML 中，每一个标签都可以称作是容器，能够放置内容。但 DIV 是 XHTML 中专门用于布局设计的容器对象。

在传统的表格布局中，完全依赖于表格对象 TABLE，在页面中绘制多个单元格，在表格中放置内容，通过表格的间距或者用无色透明的 GIF 图片来控制布局版块的间距，达到排版目的。而以 DIV 对象为核心的页面布局中，通过层来定位，通过 CSS 定义外观，最大程度地实现了结构和外观彻底分离的布局效果，因此习惯上对层布局又称为 DIV 和 CSS 布局。

1. 定义层

添加层的方法非常简单，可以通过【工具箱】面板中的 HTML 选项卡托拽一个 Div 项到设计视图中，或者在【源】视图中创建一对<div></div>标记。

【例 6-5】分析一个简单的定义 DIV 的例子。设计一个页面，添加一个层，定义其样式效果如图 6-18 所示。



图 6-18　简单的层定义

具体实现步骤如下。

(1) 在【解决方案资源管理器】中，右击网站的名称，然后单击【添加新项】，新建一个.aspx 页面，此时会发现代码中已经包含了一个层对象。

(2) 切换到【设计】视图，在【格式】菜单中单击【新建样式】命令，打开【新建样式】对话框，在【选择器】后面的文本框中输入#sample，然后选择相应的类别进行设置，完成后单击【确定】按钮。

(3) 选中层对象，在【视图】菜单中单击【管理样式】命令，然后右键单击#sample 样式，选择“应用样式”命令即可，如图 6-19 所示。

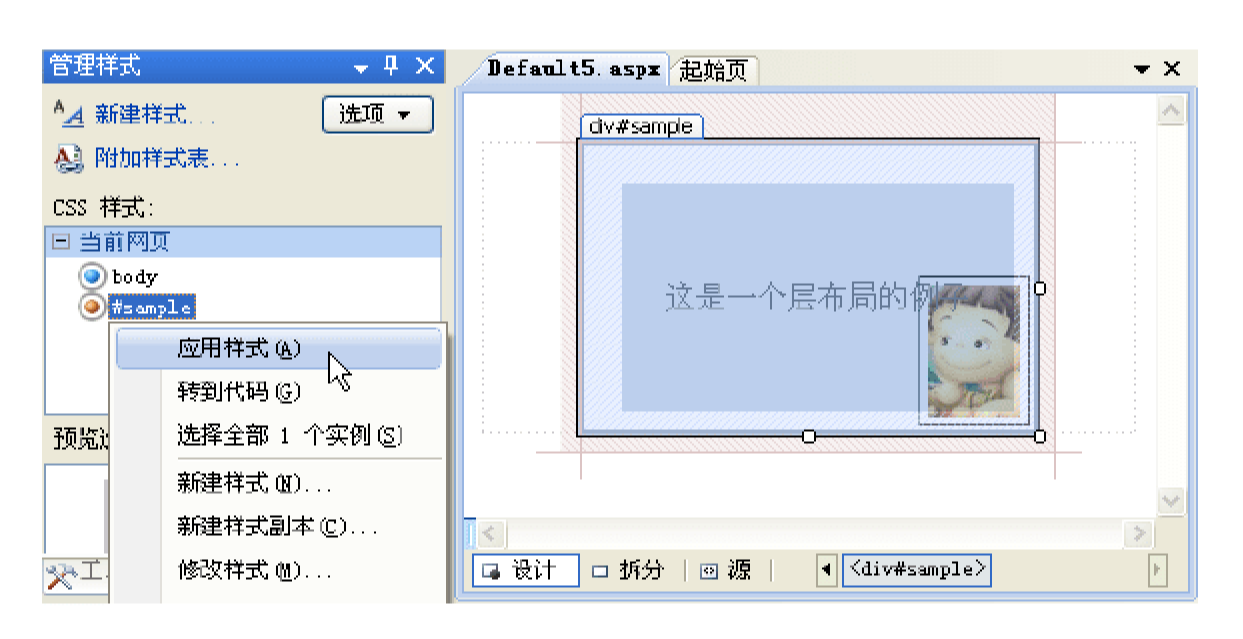


图 6-19　应用样式

对应的程序代码如下所示。

```
<head runat="server">
  <title></title>
  <style>
    body{ text-align:center; }
    #sample
    {
        margin: 10px 10px 10px 10px;
        padding:20px 10px 10px 20px;
        border-top: #CCC 2px solid;
        border-right: #CCC 2px solid;
        border-bottom: #CCC 2px solid;
        border-left: #CCC 2px solid;
        background: url(../images/bg_image1.gif) #fefefe no-repeat right bottom;
        color: #666;
        text-align: center;
        line-height: 120px;
        width:60%;
    }
  </style>
</head>
<body>
  <form id="form1" runat="server">
    <div id="sample">这是一个层布局的例子</div>
  </form>
</body>
```

其中，margin 是指层的边框以外留的空白，用于页边距或者与其他层制造一个间距。10px 10px 10px 10px 分别代表“上右下左”(顺时针方向)4 个边距。如果都一样，可以缩写成 margin: 10px;。如果边距为零，则写成 margin: 0px;。

padding 是指层的边框到层的内容之间的空白。和 margin 一样，分别指定上右下左边框到内容的距离。如果都一样，可以缩写成 padding:0px。单独指定左边可以写成 padding-left: 0px;。padding 是透明元素，不能定义颜色。

border 是指层的边框，border-right: #CCC 2px solid;是定义层的右边框颜色为#CCC，宽度

为 2px，样式为 solid 的直线。如果要虚线样式可以用 dotted。

background 是定义层的背景。分 2 级定义，先定义图片背景，采用 url(../images/bg_image1.gif)的形式来指定背景图片路径；其次定义背景色#fefefe。no-repeat 指背景图片不需要重复，如果需要横向重复用 repeat-x，纵向重复用 repeat-y，重复铺满整个背景用 repeat。后面的 right bottom;是指背景图片从右下角开始。

color 用于定义字体颜色。

text-align 用来定义层中的内容排列方式：center 居中；left 居左，right 居右。

line-height 定义行高，120px 是指绝对高度为 120 个像素，也可以写作：line-height:150%，表示是标准高度的 150%。

width 是定义层的宽度，可以采用固定值，也可以采用百分比。这个宽度仅仅指内容的宽度，不包含 margin，border 和 padding 的宽度。

2. 盒子模型

自从 1996 年 CSS1 的推出，W3C 组织就建议把所有网页上的对象都放在一个盒(box)中，设计师可以通过创建定义来控制这个盒的属性，这些对象包括段落、列表、标题、图片以及层。盒子模型主要定义 4 个区域：内容(content)、边框距(padding)、边界(border)和边距(margin)。前面【例 6-5】中定义的层就是一个典型的盒。对于初学者，经常会搞不清楚 margin，background-color，background-image，padding，content，border 之间的层次、关系和相互影响。这里给出一个盒子模型图，如图 6-20 所示，方便初学者理解和使用。

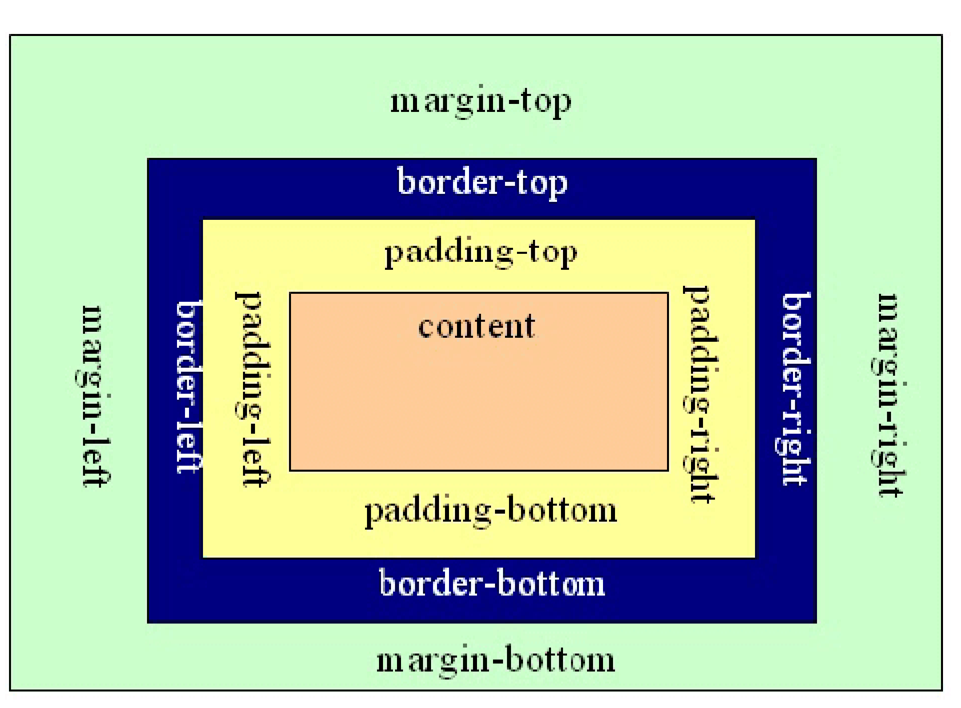


图 6-20　盒子模型

理解盒子模型就可以理解层与层之间定位的关系以及层内部的表达样式。其中，margin 属性负责层与层之间的距离，padding 属性负责内容和边框之间的距离。下面这段代码可以帮助读者进一步理解盒子模型的含义。

```
<head runat="server">
 <title></title>
 <style>
  #sample2
  {
    background-color: #FFFF00;
    border-style: solid;
```

```
        padding-bottom: 25px;
        margin-bottom: 50px;
        width: 60%;
    }
</style>
</head>
<body>
  <form id="form1" runat="server">
    <div id="sample2">W3C 组织就建议把所有网页上的对象都放在一个盒(box)中，设计师可以通过创建定义来控制这个盒的属性，这些对象包括段落、列表、标题、图片以及层</div>
    <p>这是下一段</p>
  </form>
</body>
```

运行这段代码，效果如图 6-21 所示。

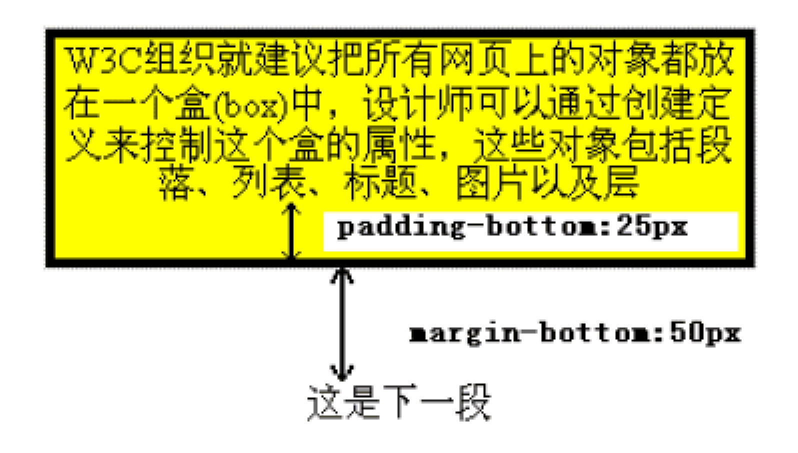


图 6-21　盒子模型举例

3. 层的定位

在一个页面中定义多个层，会发现这些层自动排列在不同的行，而要真正实现左右排列，就要加入新的属性——float(浮动属性)。float 浮动属性是 DIV 和 CSS 布局中的一个非常重要的属性。大部分的 DIV 布局都是通过 float 的控制来实现的。具体参数如下。

- float:none 用于设置是否浮动。
- float:left 用于表示对象向左浮动。
- float:right 用于表示对象向右浮动。

【例 6-6】下面通过一个左右分栏布局的例子来说明 float 的用法，该布局包含 2 个层且左右排列，这是最常用的布局结构之一，其效果如图 6-22 所示。

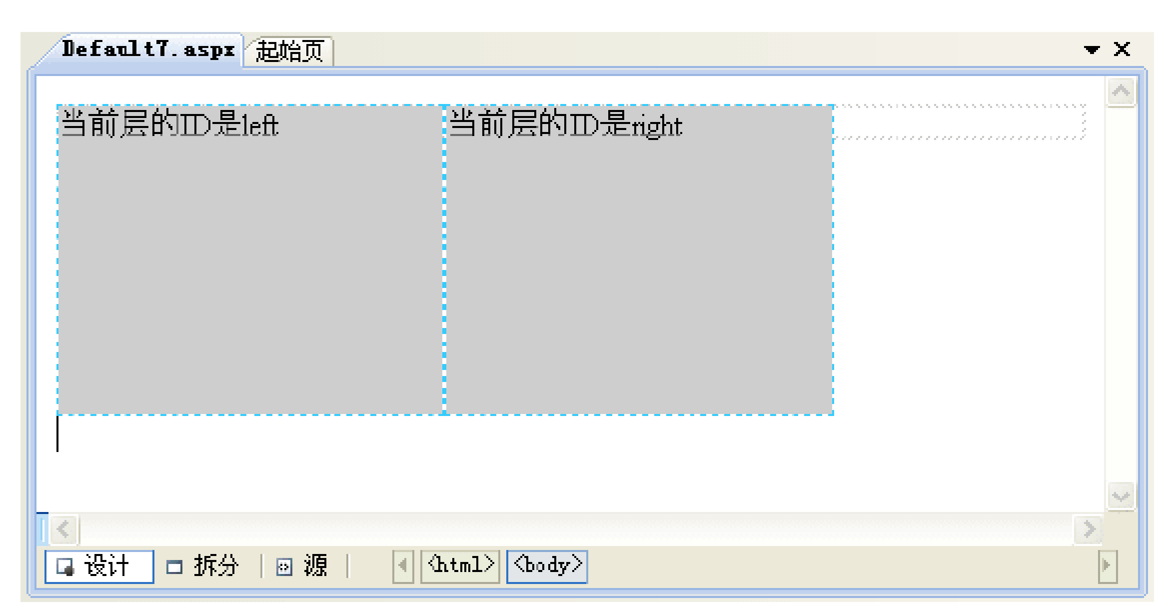


图 6-22　左右分栏效果

适应。

要想实现这样的效果，必须使用 float 属性，代码如下：

```
<head runat="server">
  <title></title>
   <style>
     #left,#right
     {
         width:200px;
         height:160px;
         background-color:#cecece;
         border:1px dashed #33ccff;
     }
     #left{float:left;}
     #right{float:left;}
   </style>
</head>
<body>
  <form id="form1" runat="server">
    <div id="left">当前层的 ID 是 left</div>
    <div id="right">当前层的 ID 是 right</div>
  </form>
</body>
```

读者可以尝试去掉#left{float:left}和#right{float:left}来看看会变成什么效果。当然，也可以把 float 的属性值改为 right 来看看会变成什么效果。

要想实现两列中左列宽度固定而右列宽度自适应窗口大小的效果，可以将【例 6-6】代码中的样式进行如下修改。

```
<style>
    #left,#right{
        background-color:#cecece;
        border:1px solid #33ccff;
        height:400px;
    }
    #left{
        width:180px;
        float:left;
    }
</style>
```

这样一来，左边的层将呈现出 180px 的宽度。而右边的层则根据浏览器窗口大小来自动适应。

还有一种左右上下分栏的样式也是非常常见的，其效果如图 6-23 所示。

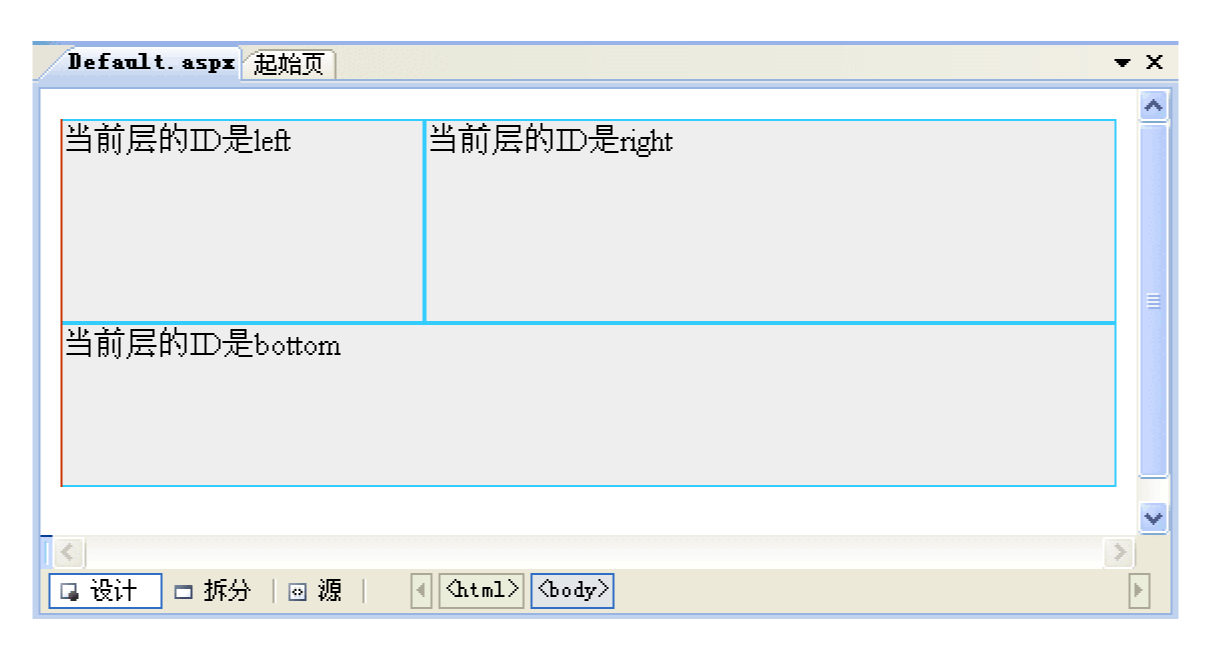


图 6-23　左右上下分栏

制作这种效果时需要将下面的层样式中添加 clear 属性，代码如下所示。

```
<head runat="server">
    <style>
     #left,#right{background-color:#eeeeee;border:1px solid #33ccff;height:200px; }
     #left{width:180px; float:left; }
     #bottom{ background-color:#eeeeee; border:1px solid #33ccff; height:50px; clear:both; }
    </style>
</head>
<body>
  <form id="form1" runat="server">
    <div id="left">当前层的 ID 是 left</div>
    <div id="right">当前层的 ID 是 right</div>
    <div id="bottom">当前层的 ID 是 bottom</div>
  </form>
</body>
```

注意：

在 IE 浏览器中，即使不定义 clear 属性为 both，依然能够按照预期的效果显示下面的层对象，但是在其他浏览器中，如果不添加这个属性，就不一定会正常显示。

4. 利用 DIV 和 CSS 实现页面布局

通过前面的介绍，可以知道 DIV 只是一个区域标识，划定了一个区域，实现样式需要借助于 CSS，这样的分离，使得 DIV 的最终效果是由 CSS 来编写的。CSS 可以实现左右分栏，也可以实现上下分栏，而表格则没有这么大的灵活性。CSS 与 DIV 的无关性，决定了 DIV 在设计上有极大的伸缩性，而不拘泥于单元格固定的模式束缚。因此，实现网页布局，通常是先在网页中将内容用 DIV 标记出来，然后再用 CSS 来编写样式。

采用 DIV 和 CSS 布局之前，首先要分析网页有哪些内容块，以及每个内容块的含义，这就是所谓的网页结构。通常情况下页面结构包含以下几块：

(1) 标题区(header)，用来显示网站的标志和站点名称等。

(2) 导航区(navigation)，用来表示网页的结构关系，如站点导航，通常放置主菜单。

(3) 主功能区(content)，用来显示网站的主题内容，如商品展示、公司介绍等。

(4) 页脚(footer)，用来显示网站的版权和有关法律声明等。

通常采用 DIV 元素来将这些结构先定义出来，类似这样：

```
<div id="header"></div>
<div id="globalnav"></div>
<div id="content"></div>
<div id="footer"></div>
```

现在还没有开始布局，这只是网页的结构，每一个内容块都可以放在页面上任何地方，放好以后就可以指定每个块的颜色、字体、边框、背景以及对齐属性等。本节将通过设计一个实际的网页，让初学者真正掌握利用 DIV 和 CSS 控制网页布局的技巧。

【例 6-7】设计一个如图 6-24 所示的计算机公司的产品销售网站的主页。



图 6-24　销售网站主页

(1) 定义网站主页的结构，根据布局，可以将网页结构分为几个部分。

```
<body>
    <form id="form1" runat="server">
    <div id="maindiv">
        <div id="HeadDiv"></div>
        <div id="MenuDiv"></div>
        <div id="ContentDiv"></div>
        <div id="EndimageDiv"></div>
        <div id="EndDiv"></div>
    </div>
    </form>
</body>
```

(2) 制作如图 6-25 所示的各种图片。



图 6-25　制作的图片

(3) 在项目中添加一个名为 StyleSheet.css 的样式表文件，然后利用样式表生成器在 body 中定义整个网页的默认字体以及网页对齐方式，代码所示如下。

```
body
{
    font-size: 12px;
    font-family: 宋体;
    text-align: center;
    background-color: #F7F7F7;
}
```

(4) 在 StyleSheet.css 文件中，定义整个网页的边距、宽度。

```
#maindiv
{
    width: 740px;
    margin-top: 0px;
    margin-left:auto;
    margin-right:auto;
}
```

(5) 在主页中添加对样式表的引用。

```
<head runat="server">
   <title>某计算机公司</title>
   <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
```

(6) 在 StyleSheet.css 文件中，定义#HeadDiv 的样式。

```
#HeadDiv
{
    width: 740px;
    background-image: url(images/logo-1.jpg);
    height: 80px;
    text-align: right;
    clip: rect(60px auto auto auto);
```

```
}
```

保存样式表文件，切换到主页的【设计】视图，显示如图 6-26 所示的效果。

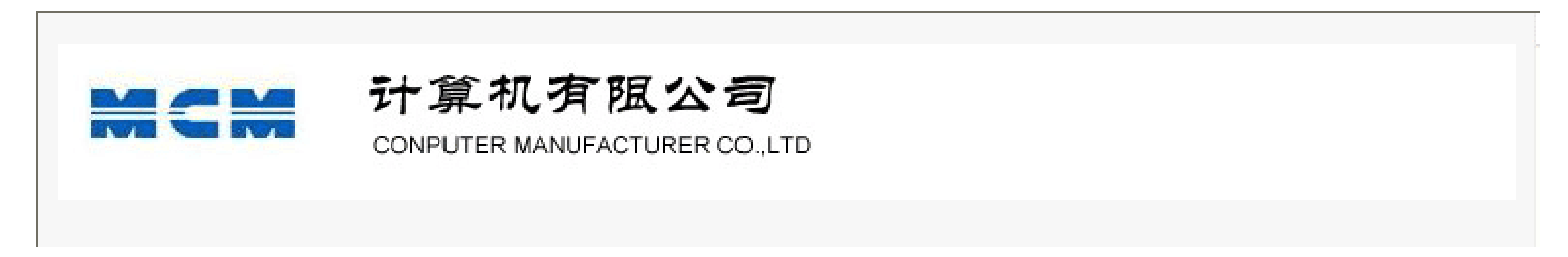


图 6-26　步骤(6)完成后的效果

(7) 在 StyleSheet.css 文件中，定义#MenuDiv 的样式。

```
#MenuDiv
{
    background-color: #3381da;
    width: 100%;
    height: 20px;
    padding-top: 6px;
    color: #ffffff;
    text-align: right;
}
#MenuDiv a, #MenuDiv a:link, #MenuDiv a:visited, #MenuDiv a:active
{
    text-decoration: none;
    color: #ffffff;
}
#MenuDiv a:hover
{
    color: #ff0000;
    text-decoration: underline;
}
```

(8) 保存样式表文件，切换到主页的【设计】视图，添加超链接，得到如下代码。

```
<div id="MenuDiv">
    |  
    <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/Default.aspx">首页
</asp:HyperLink>  |  
    <asp:HyperLink ID="HyperLink3" runat="server" NavigateUrl="#">关于公司</asp:HyperLink>
      |  
    <asp:HyperLink ID="HyperLink4" runat="server" NavigateUrl="#">新闻</asp:HyperLink>
      |  
    <asp:HyperLink ID="HyperLink5" runat="server" NavigateUrl="#">产品</asp:HyperLink>
      |  
    <asp:HyperLink ID="HyperLink6" runat="server" NavigateUrl="#">联系我们</asp:HyperLink>
      |       
</div>
```

最终得到的效果如图 6-27 所示。



图 6-27　步骤(8)完成后的效果

(9) 切换到【设计】视图，在<div id="ContentDiv"></div>层中添加 5 个层，代码如下所示。

```
<div id="ContentDiv">
    <div id="ContentLeft1">
    </div>
    <div id="ContentRight1"></div>
    <div id="ContentBottom">
        <div id="ContentLeft2"></div>
        <div id="ContentRight2"></div>
    </div>
</div>
```

然后在 StyleSheet.css 文件中，分别定义它们的样式。

```
#ContentLeft1
{
    width:368px;
    height:174px;
    background-image:url(images/news1.jpg);
    float:left;
}
#ContentRight1
{
    width:368px;
    height:174px;
    background-image:url(images/news2.jpg);
}
#ContentBottom
{
    width:740px;
    height:174px;
    height:50px; clear:both;
}
#ContentLeft2
{
    width:528px;
    height:300px;
```

```
        background-image:url(images/product.jpg);
        float:left;
    }
    #ContentRight2
    {
        width:205px;
        height:143px;
        background-image:url(images/login.jpg);
    }
```

注意，在 DIV 和 CSS 布局的过程中，也经常会使用嵌套层来配合。

(10) 在 StyleSheet.css 文件中，定义#EndimageDiv 的样式。

```
    #EndimageDiv
    {
        width: 100%;
        background-image: url(images/bg2.gif);
        background-repeat: repeat-x;
        height: 25px;
    }
```

(11) 在 StyleSheet.css 文件中，定义#EndDiv 的样式。

```
    #EndDiv
    {
        background-color: #ffffff;
        width: 100%;
        height: 63px;
        line-height: 20px;
    }
```

(12) 保存样式表文件，切换到主页的【设计】视图，添加如下代码。

```
    <div id="EndDiv">
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="#" Target="_blank">管理入口
</asp:HyperLink><br />CopyRight &copy; 2008-2009 xingkongsoft All Right Reserved.<br />
        星空软件研究室 版权所有 E-mail:xingkongsoft@163.com
    </div>
```

至此，整个页面设计结束，按 Ctrl+F5 组合键可以查看最终的效果，并进一步体会层的定位。

6.3　母版页

在 Web 站点开发中，有很多元素，如站点标题、公共导航以及版权信息等，会出现在

每一个页面中，这些元素的一致布局会让用户知道自己始终是在同一个站点中。虽然这些元素可以通过在 XHTML 中使用包含文件构建，但 ASP.NET 3.5 提供了更加健壮的母版页技术来实现。

6.3.1　母版页和内容页的概念

母版页是用于设置页面外观的模板，是一种特殊的 asp.net 网页文件，同样也具有其他 asp.net 文件的功能，如添加控件，设置样式等，只不过扩展名是.master。在母版页中，界面被分为公用区和可编辑区，公用区的设计方法与一般页面的设计方式相同，可编辑区用 ContentPlaceHolder 控件预留出来。

引用母版页的.aspx 页面称为内容页，在内容页中，母版页的 ContentPlaceHolder 控件预留的可编辑区会被自动替换为 Content 控件，开发人员只需要在 Content 控件区域中填充内容即可，在母版页中定义的其他标记将自动出现在引用该母版页的.aspx 页面中，母版页的部分以灰色显示，表示不能修改这些内容。

每一个母版页中可以包含一个或多个内容页。使用母版页可以统一管理和定义具有相同布局风格的页面，给网页设计和修改带来极大的方便。母版页的优点如下。

(1) 使用母版页可以集中处理页的通用功能，以便可以只在一个位置进行更新。

(2) 使用母版页可以方便地创建一组控件和代码，并将结果应用于一组新的页面。

(3) 通过允许控制占位符控件的呈现方式，母版页可以在细节上控制最终页的布局。

(4) 母版页提供一个对象模型，使用该对象模型可以从各个内容页自定义母版页。

在使用母版页时，需要注意的是母版页中使用的图片和超链接应尽量使用服务器端控件来实现，如 Image 和 HyperLink 控件。即使控件不需要服务器代码也是如此，这是因为将设计好的母版页或内容页移动到另一个文件夹时，如果使用的是服务器控件，即使不改变服务器控件的 URL，ASP.NET 也可以正确解析，并自动将其 URL 改为正确的位置，但是如果使用了普通 HTML 标记，那么 ASP.NET 将无法正确解析这些标记的 URL，从而导致图片不能显示和链接失败，给维护带来极大麻烦。

6.3.2　创建母版页和内容页

创建母版页的方法和创建一般页面的方法非常相似，区别是母版页无法单独在浏览器中查看，必须通过创建内容页才能浏览。下面这个例子是一个很常见的布局，母版页中包含一个标题、一个导航菜单和一个页脚，这些内容将在站点的每个页面中出现。在母版页中包含一个内容占位符，这是母版页中的一个可变区域，可以使用内容页中的信息来替换此区域。

【例 6-8】设计如图 6-28 所示的名为 Master1.Master 的母版页，然后设计两个引用母版页的内容页 Index.aspx 和 About.aspx，运行效果分别如图 6-29 和图 6-30 所示。

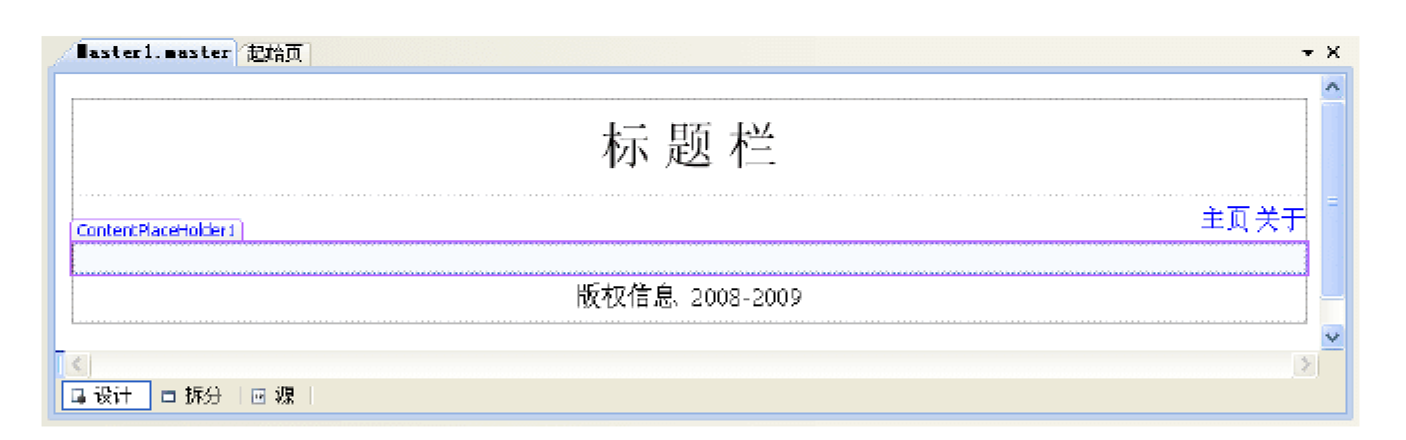


图 6-28　母版页布局

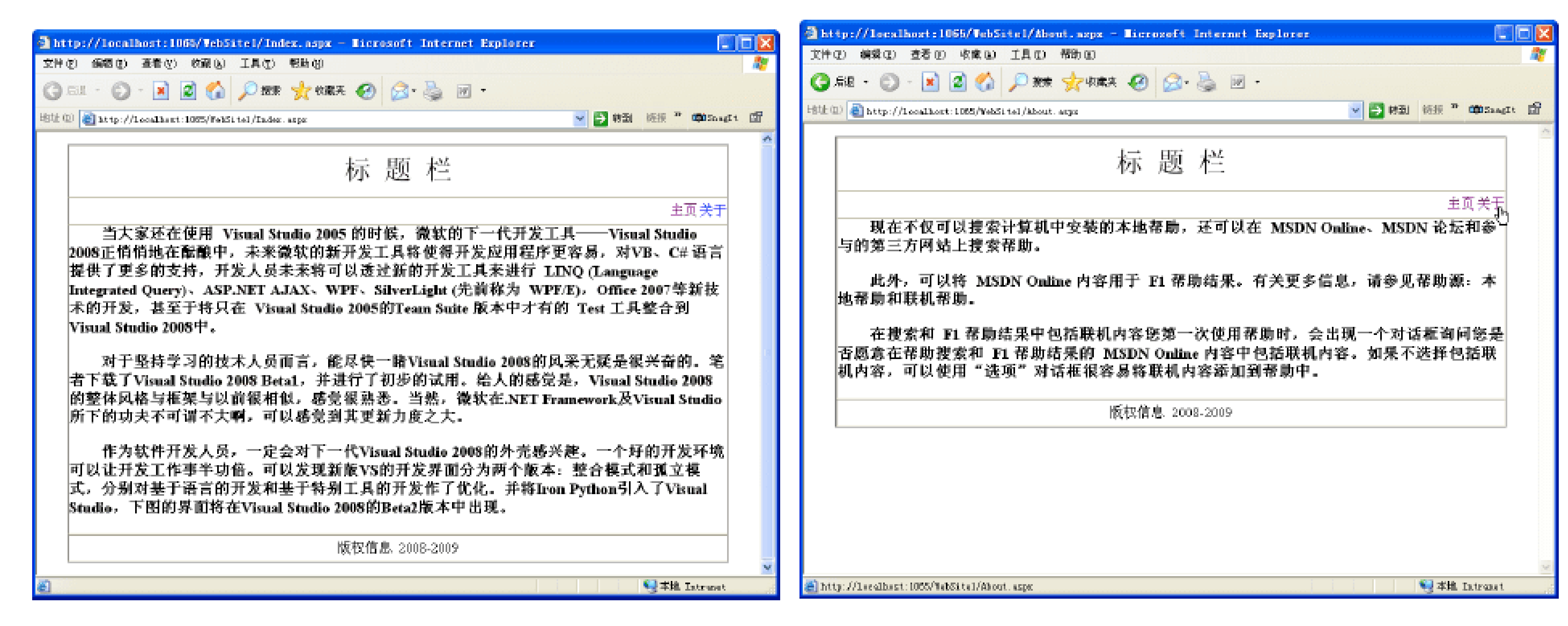


图 6-29　主页效果　　　　　　　　　图 6-30　“关于”页效果

(1) 在 VWD 2008 的【解决方案资源管理器】中右击网站的名称，然后单击【添加新项】，然后在弹出对话框中单击【母版页】。在【名称】文本框中输入 Master1，如图 6-31 所示，然后单击【添加】按钮即会在【源】视图中打开新的母版页。

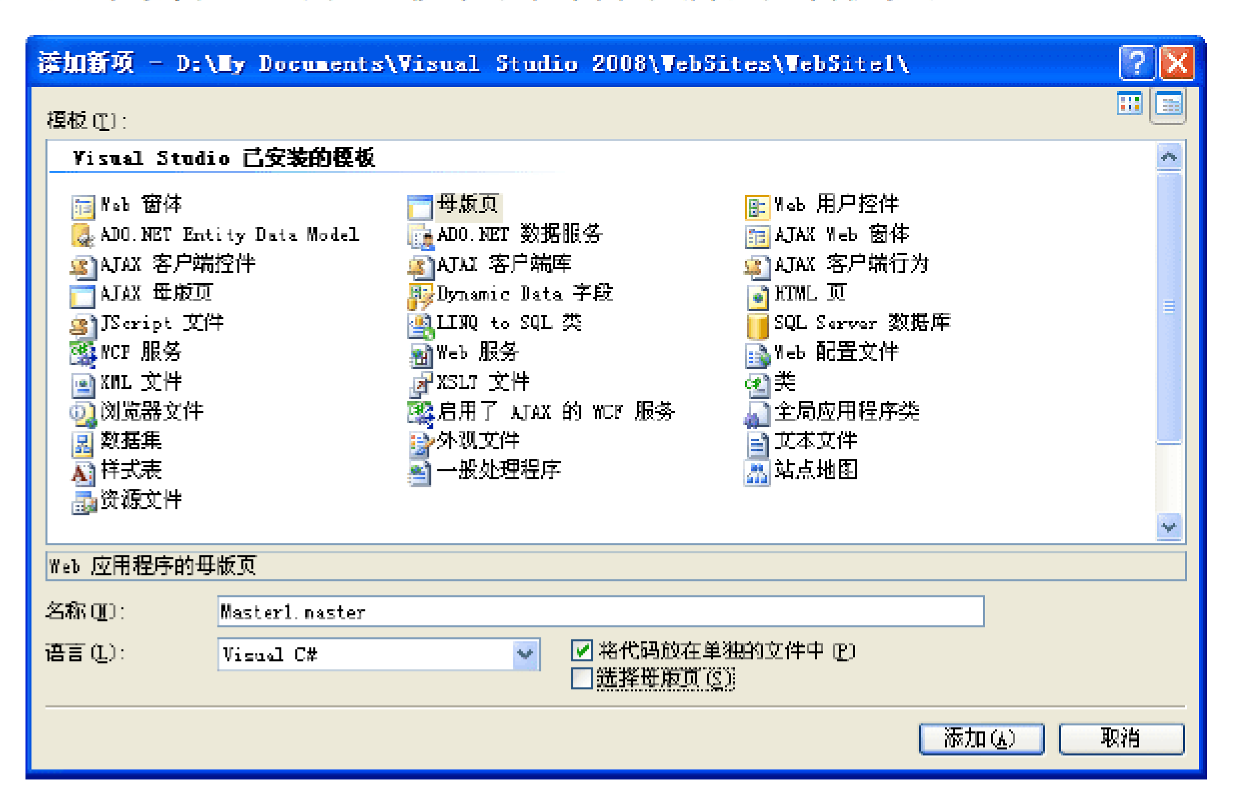


图 6-31　创建母版页

观察母版页的源代码，在页面的顶部是一个 @Master 声明，而不是通常在 ASP.NET 页顶部看到的 @Page 声明，指令如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Master1.master.cs"
Inherits="Master1" %>
```

此外，页面的主体还包含一个 ContentPlaceHolder 控件，这是母版页中的一个区域，其中的可替换内容将在运行时由内容页合并。为了方便母版页的编辑，通常情况下先将 ContentPlaceHolder 控件删除，母版页编辑完成后再放置 ContentPlaceHolder 控件，下面的步

骤将采用这种方法布局。

(2) 切换到【设计】视图，删除 ContentPlaceHolder 控件，然后单击页面中的层，插入一个 4 行 1 列的表格，边框设置为 1，表格的 width 设置为 780 像素。

(3) 布局完表格后，可以将内容添加到母版页，此内容将在所有页面上显示。例如可以在表格第 1 行添加“标题栏”，第 2 行添加 1 个 Menu 控件，第 3 行添加 1 个 ContentPlaceHolder 控件，控件的 ID 属性为 ContentPlaceHolder1，可以修改这个名字，第 4 行添加“版权信息”。其中 Menu 控件的设置内容如下：

- 将 Menu 控件的 Orientation 属性设置为 Horizontal。
- 单击 Menu 控件上的智能标记，选择【编辑菜单项…】命令，然后在【菜单项编辑器】对话框中单击【添加根项】命令图标两次，添加两个菜单项。
- 单击第一个节点，将 Text 设置为“主页”，将 NavigateUrl 设置为 Index.aspx。
- 单击第二个节点，将 Text 设置为“关于”，将 NavigateUrl 设置为 About.aspx。

然后就要为母版页添加两个带有内容的页面。第一个是主页，第二个是“关于”页面。

(4) 在【解决方案资源管理器】中右击网站的名称，然后单击【添加新项】。在弹出的对话框中单击【Web 窗体】。在【名称】文本框中输入 Index.aspx。一定要选中【选择母版页】复选框。然后单击【添加】按钮，出现【选择母版页】对话框，单击 Master1.master，然后单击【确定】按钮，创建一个新的.aspx 文件。该页面包含一个 @ Page 指令，此指令将当前页附加到带有 Master1 属性的选定母版页，如下面的代码示例所示。

```
<%@ Page Language="C#" MasterPageFile="~/Master1.master" ... %>
```

(5) 切换到【设计】视图。母版页中的 ContentPlaceHolder 控件在新的内容页中显示为 Content 控件。而其他的母版页内容显示为浅灰色，表示在编辑内容页时不能更改这些内容。在与母版页上的 ContentPlaceHolder1 匹配的 Content 控件中，输入主页要显示的内容，然后选择文本，通过从【工具箱】上的【块格式】下拉列表中选择【标题 2】，保存页面。

(6) 使用同样的方法创建“关于”内容页，名字为 About.aspx。

(7) 设置 Index.aspx 为起始页，按 Ctrl+F5 组合键运行并测试这个站点。ASP.NET 将 Index.aspx 页的内容与 Master1.master 页的布局合并到单个页面，并在浏览器中显示产生的页面。需要注意的是，此页的 URL 为 Index.aspx，浏览器中是不存在对母版页的引用的。单击“关于”链接，显示 About.aspx 页，它也是和 Master1.master 页合并的结果。具体例子参见随书附带的源程序。

6.3.3　从内容页访问母版页的成员

利用内容页的后台代码可以引用母版页上的成员，包括母版页上的任何公共属性或方法以及任何控件。要实现内容页对母版页中定义的属性或方法进行访问，则该属性或方法必须声明为公共成员(public)，也可以对母版页动态地进行访问。

1. 访问母版页的公共成员

要想在内容页中访问母版页上的属性，必须在母版页上先创建一个属性，创建的方法是：

(1) 切换到或打开 Master1.master 页，在【解决方案资源管理器】中右击 Master1.master，然后单击【查看代码】命令，打开代码编辑器。

(2) 在类定义中输入以下代码。

```
public String WebSiteName
{
  get { return (String) ViewState["websiteName"]; }
  set { ViewState["websiteName"] = value; }
}
```

此代码为母版页创建名为 WebSiteName 的属性。在视图状态中存储此值，以便此值在访问期间保持不变。

(3) 在类定义中再添加以下代码。

```
void Page_Init(Object sender, EventArgs e)
{
  this.WebSiteName = "母版页的介绍";
}
```

接下来修改内容页，通过引用 WebSiteName 属性，来使用母版页的公共成员。

(4) 切换到或打开 Index.aspx 页，并切换到该页的【源】视图。在页面顶部的 @ Page 指令下，添加下面的 @ MasterType 指令。

```
<%@ MasterType virtualpath="~/Master1.master" %>
```

此指令的作用是将内容页的 Master 属性绑定到 Master1.master 页。

(5) 切换到【设计】视图，在 Content 控件中，增加一行内容“该网站是关于”。

(6) 从【工具箱】中将 Label 控件拖动到 Content 控件上，并将其放置到静态文本的后面，使文本如下所示。

```
该网站是关于 [Label]
```

(7) 在【解决方案资源管理器】中右击 Index.aspx，然后单击【查看代码】打开代码编辑器。在类定义中添加下面的代码。

```
void Page_Load(Object sender, EventArgs e)
{
   Label1.Text = Master.WebSiteName;
}
```

(8) 测试内容页，切换到或打开 Index.aspx 页，然后按 Ctrl+F5 组合键运行页面。页面即会在浏览器中显示，其中文本为“该网站是关于 母版页的介绍”。

(9) 修改母版页中属性的值，重新运行页面将看到新的属性值显示在页面中。

2. 动态访问母版页

在有些情况下，可能希望能够动态更改母版页。也就是使用代码设置内容页的母版页。例如，可能希望允许用户从几个布局中进行选择，根据个人喜好设置母版页。

首先，设计者要保证网站中有不少于 2 个的母版页，然后创建按钮使用户能够在两个母版页之间切换。在本例中，新创建的第 2 个母版页和第 1 个母版页非常相似，只是将标题栏和版权信息的字体改了一下，具体步骤如下。

(1) 切换到或打开 Master1.master 页。在【工具箱】中将 LinkButton 按钮控件拖动到页面上，并将其放置在标题栏内容的右边。

(2) 将按钮控件的 Text 属性设置为“变换字体效果”，如图 6-32 所示。

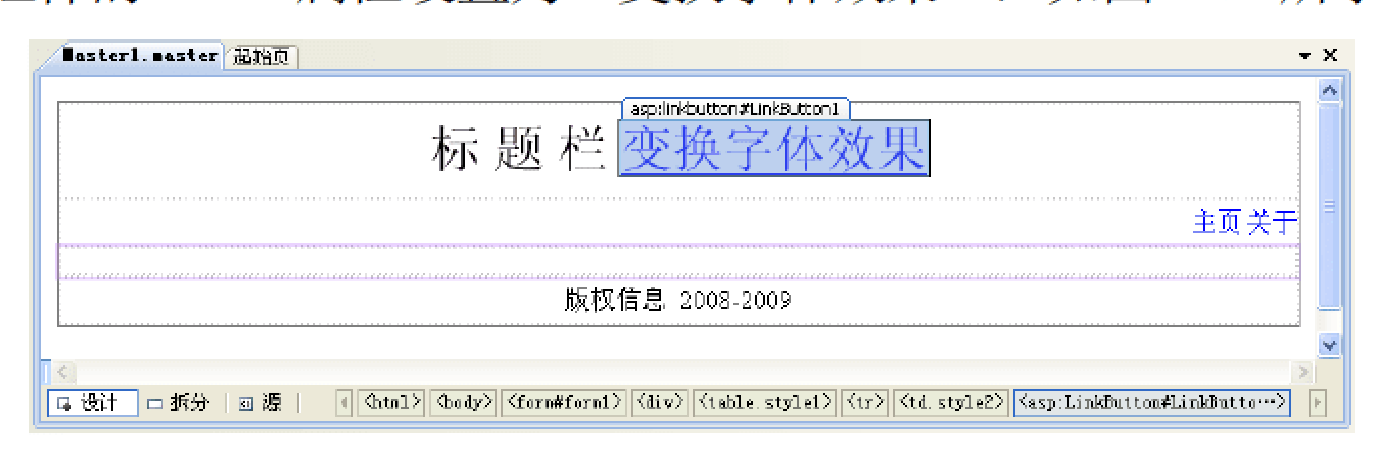


图 6-32　修改后的 Master1 的设计界面

(3) 双击此按钮，为其 Click 事件创建一个处理程序，然后添加以下代码。

```
void LinkButton1_Click(Object sender, EventArgs e)
{
    Session["masterpage"] = "Master2.master";
    Response.Redirect(Request.Url.ToString());
}
```

此代码将第 2 个母版页的文件名加载到一个持久的会话变量中，然后重新加载当前页。

(4) 创建第 2 个母版页，内容和 Master1.master 完全相同，只是字体效果不同，和 Master1 一样，添加 1 个 LinkButton 控件，并将其 Text 属性设置为“返回正常效果”。如图 6-33 所示。



图 6-33　Master2 的效果

(5) 双击此按钮，为其 Click 事件创建一个处理程序，然后添加下面的代码。

```
void LinkButton1_Click(Object sender, EventArgs e)
{
    Session["masterpage"] = "Master1.master";
    Response.Redirect(Request.Url.ToString());
}
```

此代码与 Master1.master 页中按钮的代码类似，不同之处在于它加载第一个母版页。最后一步是在内容页中编写代码，此代码将动态加载用户选定的母版页。

(6) 切换到或打开 About.aspx 页。在【解决方案资源管理器】中右击 About.aspx，然后单击【查看代码】命令，打开代码编辑器。在类定义中添加下面的代码。

```
void Page_PreInit(Object sender, EventArgs e)
{
    if(Session["masterpage"] != null)
    {
        this.MasterPageFile = (String) Session["masterpage"];
    }
}
```

此代码将当前页的 MasterPageFile 属性的值设置为会话变量中的值(如果有)。此代码必须在 Page_PreInit 处理程序中运行，因为必须建立母版页，使得页面可以创建其实例，然后可以进一步初始化。

(7) 测试动态母版页。在 About.aspx 页中按 Ctrl+F5 组合键运行此页面。此页即会在浏览器中显示，它与其默认母版页 Master1.master 合并。单击“变换字体效果”链接，将会重新显示此页，但这一次它是与 Master2.master 合并。

6.4 主题

网站的美观主要涉及页面和控件的样式属性，在 ASP.NET 应用程序中，可以利用 CSS 控制页面上各元素的样式以及部分服务器控件的样式，但是，有些服务器控件的属性无法通过 CSS 进行控制。为了解决这个问题，从 ASP.NET 2.0 开始就提供了一种称为“主题”的新方式，它可以保持网站外观的一致性和独立性，同时使页面的样式控制更加灵活方便，例如动态实现不同用户界面的切换等。ASP.NET 3.5 继承了这个特性。

6.4.1 主题的概念

主题是指页面和控件外观属性设置的集合。主题由一个文件组构成，包括皮肤文件(扩展名为.skin)、级联样式表文件(扩展名为.css)、图片和其他资源等的组合，但一个主题至少包含一个皮肤文件。

主题分为两大类型，一类是应用程序主题，另一类是全局主题。

应用程序主题是指保存在 Web 应用程序的 App_Themes 文件夹下的一个或多个主题文件夹，主题的名称就是文件夹的名称。

全局主题是指保存在服务器上，根据不同的服务器配置决定的，能够对服务器上所有 Web 应用程序起作用的主题文件夹。

一般情况下，很少用到全局主题，而本书所讲的主题仅指应用程序主题，即保存在应用程序中 App_Themes 文件夹下的主题文件夹，简称主题。

打开一个 Web 应用程序，在【解决方案资源管理器】中，右击项目名，选择【添加】→【添加 ASP.NET 文件夹】→【主题】命令，系统自动生成 App_Themes 文件夹，并在该文件夹下生成一个默认名为“主题”的文件夹。在 App_Themes 文件夹中可以创建多个主题，方法相同。

1. 皮肤文件

皮肤文件是主题的核心文件，也称为外观文件，专门用于定义服务器控件的外观。在主题中可以包含一个或多个皮肤文件，后缀名为.skin。

在控件皮肤设置中，只能包含主题的属性定义，如样式属性、模板属性、数据绑定表达式等，不能包含控件的 ID，如 Label 控件的皮肤设置代码如下：

```
<asp:Label runat="server" BackColor="Blue" Font-Names="Arial Narrow" />
```

这样一旦将该皮肤应用到 Web 页面中，则所有的 Label 控件都将显示皮肤所设置的样式。

右击某一个“主题”文件夹，选择【添加新项】命令，在弹出对话框中选择【外观文件】，并在【名称】文本框中修改皮肤文件名，单击【添加】按钮即可添加一个皮肤文件。同样的方法可以添加多个皮肤文件。

2. 级联样式表文件

主题中可以包含一个或多个 CSS 文件，一旦 CSS 文件被放在主题中，则应用时无需再在页面中指定 CSS 文件链接，而是通过设置页面或网站所使用的主题即可，当主题得到应用时，主题中的 CSS 文件会自动应用到页面中。

右击某一个“主题”文件夹，选择【添加新项】命令，在弹出的对话框中选择【样式表文件】，并在【名称】文本框中修改样式表文件名，单击【添加】按钮即可添加一个样式表文件。同样的方法可以添加多个样式表文件。

图 6-34 中创建了 3 个主题，分别是 Theme1、Theme2 和 Theme3，Theme1 中包含 1 个皮肤文件，Theme2 中包含 2 个皮肤文件，Theme3 中包含 1 个皮肤文件和 1 个样式表文件。

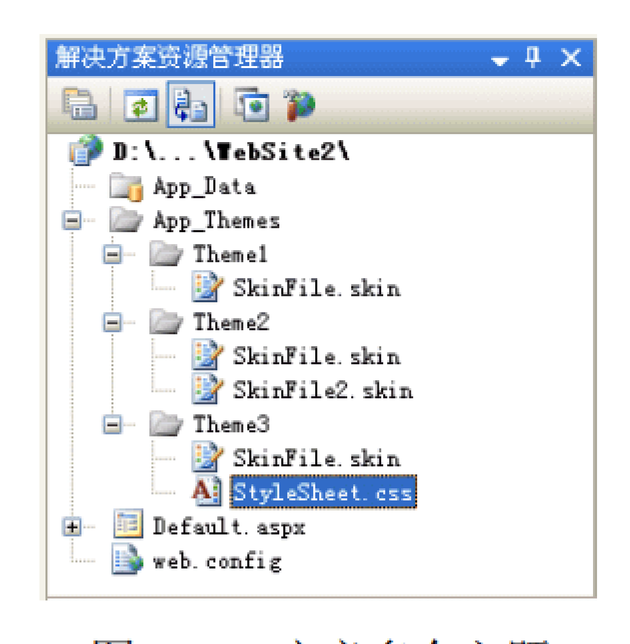


图 6-34　定义多个主题

6.4.2　在主题中定义外观

ASP.NET 使得将预定义的主题应用于页或创建唯一的主题变得很容易。下面通过一个简单的例子来说明外观定义的方法。

【例 6-9】创建一个包含一些简单外观的主题，这些外观用于定义控件的外观。

(1) 在 VWD 2008 中，右击网站名，单击【添加 ASP.Net 文件夹】，然后单击【主题】。将创建名为 App_Themes 的文件夹和名为“主题”的子文件夹。将【主题】文件夹重命名为 Theme1。此文件夹名将成为创建的主题的名称。

(2) 右击 Theme1 文件夹，选择【添加新项】，添加一个新的外观文件，然后将该文件命名为 sampleSkin.skin。在 sampleSkin.skin 文件中，按下面的代码示例所示的方法添加外观定义。

```
<asp:Label runat="server" ForeColor="red" Font-Size="14pt" Font-Names="Verdana" />
<asp:button runat="server" Borderstyle="Solid" Borderwidth="2px" Bordercolor="Blue" Backcolor="yellow"/>
```

外观定义与创建控件的语法类似，不同之处在于，定义只包括影响控件外观的设置，不包括 ID 属性的设置。

(3) 保存该外观文件。

(4) 新建一个网页文件，切换到【设计】视图中，添加 1 个标签控件和 1 个按钮控件，具体位置无所谓，如图 6-35 所示。

(5) 在属性面板中选择 Document 元素，设置 Theme 属性的值为 Theme1，切换到【源】视图中，会发现代码第 1 行的@ Page 指令中添加了下面的属性：

```
<%@ Page … Theme="Theme1"%>
```

(6) 保存文件，按 Ctrl+F5 组合键执行该页面查看设置效果，如图 6-36 所示。

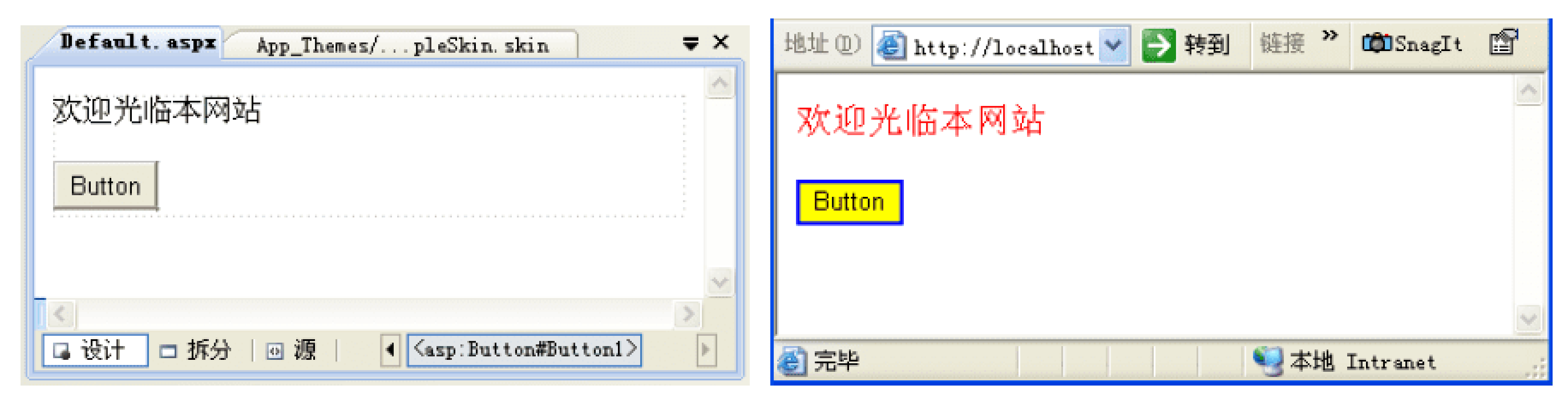


图 6-35　设置外观前　　　　图 6-36　设置外观后

在该网页文件中，将该主题设置成另一个主题(如果存在)的名称。再按 Ctrl+F5 组合键再次运行该页。控件将再次更改外观。

在皮肤文件中，系统没有提供控件属性设置的智能提示功能，所以一般不在皮肤文件中直接编写定义控件外观的代码，而是先在页面中设置控件的属性，然后再将自动生成的代码复制到外观文件中进行修改。因此，【例 6-9】也可以这样来实现。

(1) 创建一个 Web 页面，添加相应的控件并设置其外观。

(2) 新建一个主题，将相应控件的源代码复制到该主题的皮肤文件中，并去掉所有控件的 ID 属性。

(3) 在其他页面的属性面板中选择 Document 元素，设置 Theme 属性的值为相应的主题即可。

如果希望某些控件的外观和页面中具有相同类型的其他控件的外观不一样，则可以通过

在.skin 文件中给特定的控件添加一个 SkinID 属性，例如，在【例 6-9】中增加一个按钮，其外观定义成如下样式：

```
<asp:Button runat="server" SkinID="GreenButton" Borderstyle="dotted" Borderwidth="2px"
Bordercolor="red" Backcolor="Green"/>
```

修改按钮控件的 SkinID 属性的值为 GreenButton。这样，新增加的按钮就和原来的按钮显示了不同的外观。

6.4.3　在主题中同时定义外观和样式表

【例 6-9】只定义了一个皮肤文件，实际上，在主题中还可以定义 .css 文件。要想让定义的 .css 文件起作用，需要在网页文件中设置 StyleSheetTheme 属性为定义的主题即可。

【例 6-10】演示如何在网页文件中同时使用皮肤文件和样式表文件。

(1) 在 VWD 2008 中，右击网站名，单击【添加 ASP.Net 文件夹】，然后单击【主题】。将创建名为 App_Themes 的文件夹和名为“主题”的子文件夹。将【主题】文件夹重命名为 Theme2。此文件夹名将成为创建的主题的名称。

(2) 右击 Theme2 文件夹，选择【添加新项】，添加一个新的外观文件，然后将该文件命名为 Skin1.skin。在 Skin1.skin 文件中，将网页文件中要用到的所有控件的外观定义添加进来，注意不能含有任何控件的 ID，外观代码如下所示：

```
<asp:Label runat="server" BackColor="#FFFFCC" BorderColor="#6600FF"
  BorderStyle="Solid" BorderWidth="4px" Font-Bold="True" Font-Names="华文彩云"
  Font-Size="XX-Large" ForeColor="#CC0099" style="text-align: center" Width="206px">
</asp:Label>
<asp:Button runat="server" BackColor="#3333CC" BorderColor="#000099"
Font-Bold="True" Font-Size="Medium" ForeColor="White"/>
<asp:TextBox runat="server" BackColor="#99FFCC" Columns="10"></asp:TextBox>
```

(3) 在主题 Theme2 的文件夹下，再添加一个 Stylesheet1.css 的样式文件，文件内容如下所示：

```
.style1    /*  用于修饰表格 */
{
    width: 200px;
    border-collapse: collapse;
    border: 1px solid #800080;
}
.style2    /*  用于修饰单元格 */
{
    font-family: 幼圆;
    font-size: large;
    font-weight: bold;
}
```

(4) 新建一个 Web 页面，切换到【设计】视图下添加表格和相应的控件，其最终效果如图 6-37 所示。

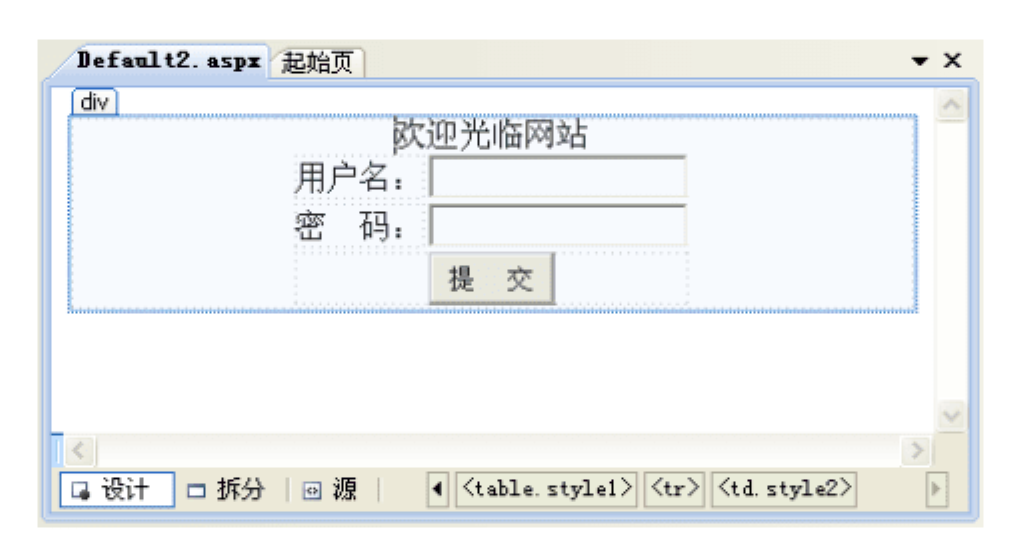


图 6-37　没有引入皮肤和样式前的效果

其对应的代码请参见随书附带的源程序。

修改当前页面的 Document 中属性 StyleSheetTheme 的值为 Theme2，即可看到引入皮肤和样式表文件后的最终显示效果，如图 6-38 所示。

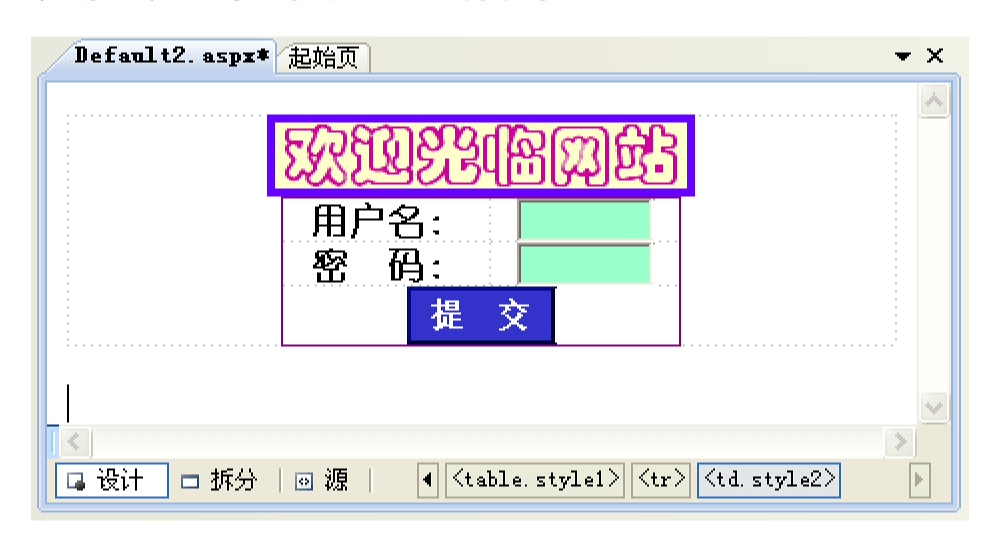


图 6-38　引入皮肤和样式后的效果

注意，创建了主题后，可以定制如何在应用程序中使用主题，方法是：将主题作为自定义主题与网页文件关联，或者将主题作为样式表主题与网页文件关联。样式表主题和自定义主题都使用相同的主题文件，但是样式表主题在网页文件的控件和属性中的优先级最低。在 ASP.NET 中，优先级的顺序是：

(1) 主题设置，包括 Web.config 文件中设置的主题。

(2) 本地网页文件的样式属性设置。

(3) 样式表主题设置。

在这里，如果选择使用样式表主题，则在网页文件中本地声明的任何样式信息都将覆盖样式表主题的属性。同样，如果使用自定义主题，则主题的属性将覆盖本地网页文件中设置的任何样式内容，以及使用中的任何样式表主题中的任何内容。

6.4.4　利用主题实现换肤

前面介绍的内容只是有关如何指定页面主题的方法，并没有涉及后台代码。但在实际使用中，可能需要通过后台代码来控制主题的指定，例如通过按钮来为页面指定不同的主题以实现换肤的效果，像很多论坛、博客系统都具有类似功能。

【例 6-11】介绍在创建外观定义时使用设计器来设置外观属性，然后将控件定义复制到外观文件，通过基于现有控件来创建自定义主题，这是非常简单的一种方法。

【例 6-11】利用主题实现换肤功能。

(1) 在 VWD 2008 中，新创建一个网站，右击网站名，单击【添加 ASP.Net 文件夹】，然后单击【主题】。将创建名为 App_Themes 的文件夹和名为“主题”的子文件夹。将【主题】文件夹重命名为 Theme1。此文件夹名将成为创建的第 1 个主题的名称。同样的方法创建第 2 个主题，重命名为 Theme2。

(2) 右击 Theme1 文件夹，选择【添加新项】，添加一个新的外观文件，重命名为 Skin1.skin。同样的方法在 Theme2 文件夹中添加一个新的外观文件，重命名为 Skin2.skin。

(3) 新建一个 Web 页面，切换到【设计】视图，添加 2 个“日历”控件，设置第 1 个“日历”控件的自动套用格式为“彩色型 1”，设置第 2 个“日历”控件的自动套用格式为“彩色型 2”。切换到【源】视图，并复制这两个<asp:calendar>元素及其属性，分别粘贴到 Skin1.skin 文件中和 Skin2.skin 文件中。然后删除每一个<asp:calendar>元素中的 ID 属性。保存这两个皮肤文件。

(4) 切换到 Default.aspx 页面，将一个“日历”控件拖到页上。不要设置该控件的任何属性。然后再添加两个“按钮”控件，效果如图 6-39 所示。



图 6-39　没有引入主题的日历

(5) 双击【彩色型 1】按钮，为其 Click 事件创建一个处理程序，然后添加以下代码。

```
void Button1_Click(Object sender, EventArgs e)
{
    Session["themepage"] = "Theme1";   // 将彩色型 1 主题保存到 Session 中
    Response.Redirect(Request.Url.ToString());
}
```

(6) 双击【彩色型 2】按钮，为其 Click 事件创建一个处理程序，然后添加以下代码。

```
void Button2_Click(Object sender, EventArgs e)
{
    Session["themepage"] = "Theme2";    // 将彩色型 2 主题保存到 Session 中
    Response.Redirect(Request.Url.ToString());
}
```

(7) 在类定义中再添加下面的代码。

```
void Page_PreInit(object sender, EventArgs e)
{
     if (Session["themepage"] != null)
```

```
        {
            this.Theme = (String)Session["themepage"];
        }
        else
        {
            this.Theme = "Theme1";    // 默认加载彩色型 1 主题
        }
    }
```

(8) 测试动态主题页。在 Default.aspx 页中按 Ctrl+F5 组合键运行此页面。此页即会在浏览器中显示，默认以彩色型 1 主题显示日历，如图 6-40 所示。单击 【彩色型 2】按钮，以彩色型 2 主题显示日历，如图 6-41 所示。单击不同的按钮就可以实现换肤的效果了。



图 6-40　彩色型 1 主题　　　图 6-41　彩色型 2 主题

前面的内容都是将主题应用于某一个页面文件，其实还可以将一个主题应用于整个网站。为网站设置主题的方法如下所示。

(1) 打开网站的配置文件 Web.config，在<system.web>元素内部添加<pages>元素。

(2) 将下列属性添加到<pages>元素中：<pages theme="sampleTheme" />。

(3) 保存并关闭 Web.config 文件。

(4) 打开所有包含主题的页面，并切换到【源】视图。从页声明中移除 theme="themeName" 属性。

此后，网站中所有页面都将使用 Web.config 文件中指定的主题显示。如果选择在某一个页面文件的声明中再指定一个主题名称，则该主题名称将覆盖 Web.config 文件中指定的任何主题。

实验 6　页面布局

1. 实验目的

熟悉层的应用，掌握层的布局；掌握母版页的创建及使用方法。能够通过创建母版页来

实现导航页面的设计。

2. 实验内容和要求

(1) 新建一个名为 CRM 的网站。

(2) 在【解决方案资源管理器】中右击网站的名称，然后单击【添加新项】，然后在弹出的对话框中单击【母版页】。在【名称】文本框中输入 Master1，然后单击【添加】按钮即会在【源】视图中打开新的母版页。

(3) 切换到【设计】视图，删除 ContentPlaceHolder 控件，然后插入 4 个层，代码如下所示。

```
<div id="top"></div>
<div id="left">
        <asp:HyperLink ID="hpl_CNotify" runat="server" NavigateUrl="~/Module/CNotify.aspx"
Target="_self">公告信息</asp:HyperLink>
        <asp:HyperLink ID="hpl_CSearch" runat="server" NavigateUrl="~/Module/CSearch.aspx"
Target="_self">资料查询</asp:HyperLink>
        <asp:HyperLink ID="hpl_CAdd" runat="server" NavigateUrl="~/Module/CAdd.aspx"
Target="_self">资料添加</asp:HyperLink>
        <asp:HyperLink ID="hpl_CManage" runat="server" NavigateUrl="~/Module/CManage.aspx"
Target="_self">资料管理</asp:HyperLink>
        <asp:HyperLink ID="hpl_Exit" runat="server" NavigateUrl="~/Module/Exit.aspx">退出系统
</asp:HyperLink>
 </div>
 <div id="right">
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
            </asp:ContentPlaceHolder>
</div>
<div id="bottom">版权所有，违者必究  </div>
```

(4) 分别设置每个层的 CSS 样式，代码大致如下所示。

```
#left,#right{border:0px solid;float:left}
#left{width:160px;height:450px}
#top{ border:0px solid; height:120px;clear:both;}
#bottom{    border:0px solid; height:50px; clear:both; }
```

(5) 最后还需要进一步的详细设置，最终效果如图 6-42 所示。

汽车客户关系管理系统

公告信息
资料查询
资料添加
资料管理
退出系统

版权所有，违者必究

图 6-42　最终效果

第7章　ADO.NET数据访问

ASP.NET 应用程序的数据访问是通过 ADO.NET 进行的，ADO.NET 可以使 Web 应用程序从各种数据源中快速访问数据。从传统的数据库到 XML 数据存储，各种各样的数据源都能连接到 ADO.NET，从而更加灵活地访问数据，减少访问数据所需的代码，提高了开发效率和 Web 应用程序的性能。

本章首先介绍 ADO.NET 的基本知识，然后再详细介绍在 ASP.NET 中的几种数据访问方法，而有关数据绑定的内容则放到第 8 章再进行介绍。

本章的学习目标

- 了解 ADO.NET 的基本知识
- 掌握 ADO.NET 与数据库的连接方法
- 掌握利用 Command 访问数据库的方法
- 掌握利用 DataAdapter 访问数据库的方法

7.1　ADO.NET 概述

ADO.NET 是.NET Framework 提供的数据访问的类库，ADO.NET 对 Microsoft SQL Server、Oracle 和 XML 等数据源提供一致的访问。应用程序可以使用 ADO.NET 连接到这些数据源，并检索和更新所包含的数据。

ADO.NET 用于数据访问的类库包含.NET Framework 数据提供程序和 DataSet 两个组件。.NET Framework 数据提供程序与 DataSet 之间的关系如图 7-1 所示。

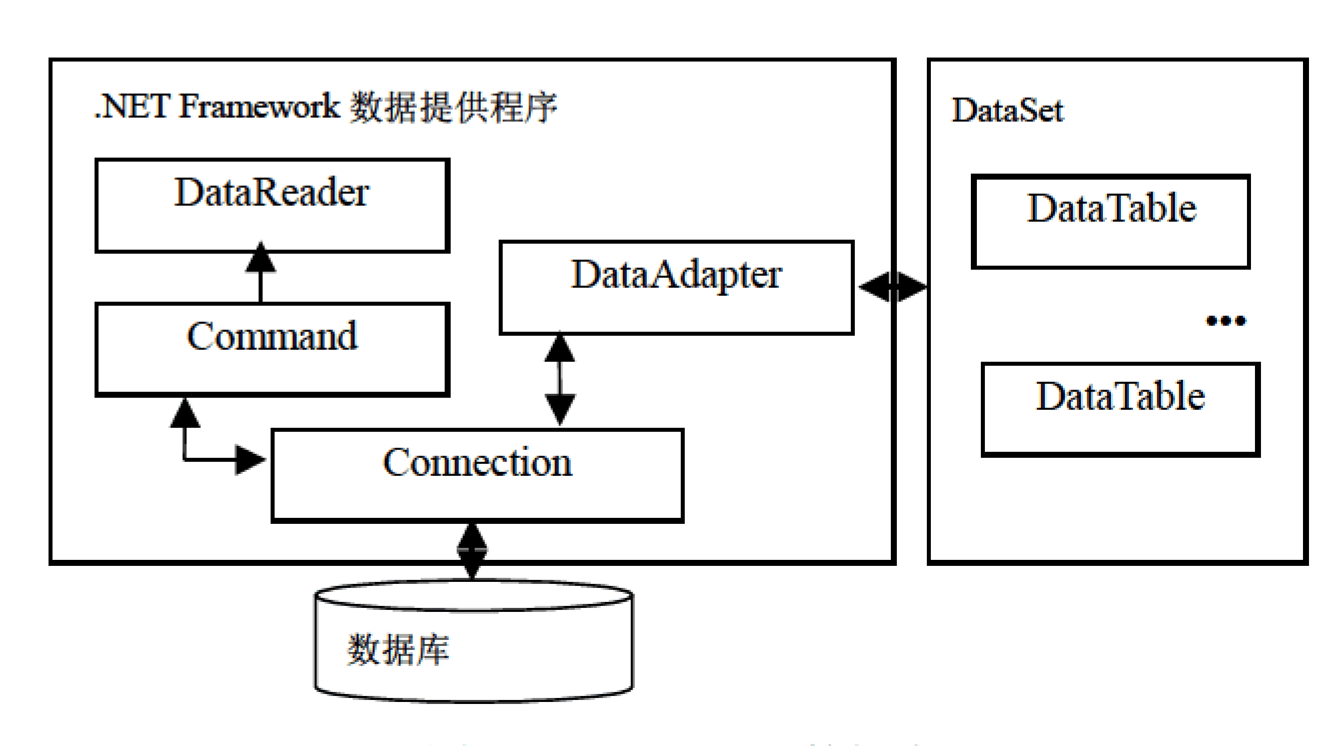


图 7-1　ADO.NET 的组成

.NET Framework 数据提供程序包含以下 4 个核心类。

- Connection：建立与数据源的连接。
- Command：对数据源执行操作命令，用于修改数据、查询数据和运行存储过程等。
- DataReader：从数据源获取返回的数据。
- DataAdapter：用数据源数据填充 DataSet，并可以处理数据更新。

DataSet 是 ADO.NET 的断开式结构的核心组件。设计 DataSet 的目的是为了实现独立于任何数据源的数据访问，可以把它看成是内存中的数据库，是专门用来处理数据源中读出的数据。

DataSet 的优点就是离线式，一旦读取到数据库中的数据后，就在内存中建立数据库的副本，在此之后的操作，直到执行更新命令为止，所有的操作都是在内存中完成的。不管底层的数据库是哪种形式，DataSet 的行为都是一致的。

DataSet 是数据表(DataTable)的集合，它可以包含任意多个数据表，而且每个 DataSet 中的数据表对应一个数据源中的数据表(Table)或者是数据视图(View)。DataSet 的结构如图 7-2 所示。

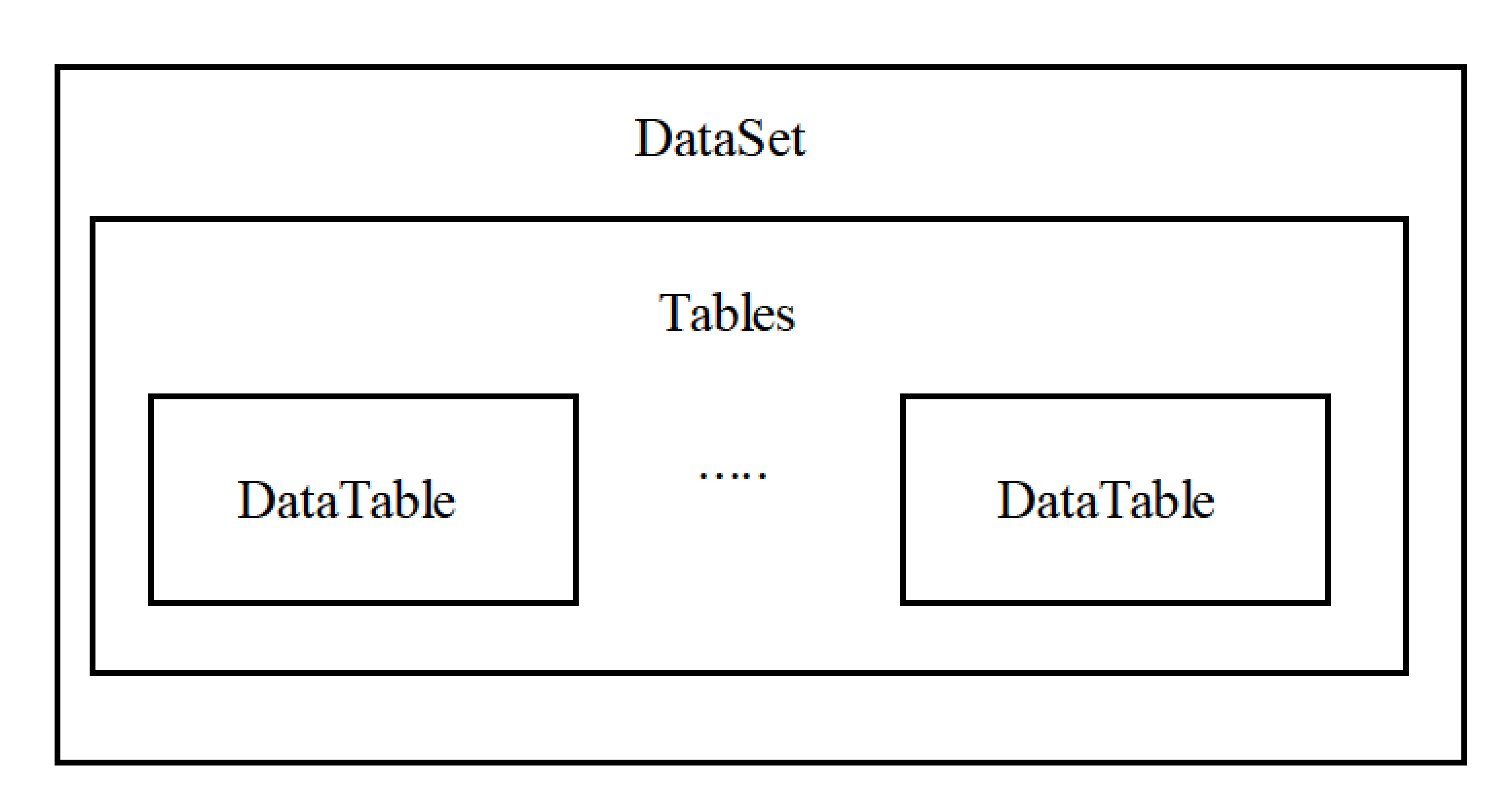


图 7-2　DataSet 的结构

ASP.NET 数据访问程序的开发流程有以下几个步骤：

(1) 利用 Connection 对象创建数据连接。

(2) 利用 Command 对象数据源执行 SQL 命令。

(3) 利用 DataReader 对象读取数据源的数据。

(4) DataSet 对象与 DataAdapter 对象配合，完成数据的查询和更新操作。

在 ADO.NET 中，连接数据源有 4 种接口：SQLClient、OracleClient 、ODBC、OLEDB。其中 SQLClient 是 Microsoft SQL Server 数据库专用连接接口，OracleClient 是 Oracle 数据库专用的连接接口，ODBC 和 OLEDB 可用于其他数据源的连接。在应用程序中使用任何一种连接接口时，必须在后台代码中引用对应的名称空间，类的名称也随之发生变化，如表 7-1 所示。

表 7-1　数据连接方式名称空间与对应的类名称

名 称 空 间	对应的类名称
System.Data.SqlClient	SqlConnection；SqlCommand；SqlDataReader；SqlDataAdapter
System.Data.Odbc	OdbcConnection；OdbcCommand；OdbcDataReader；OdbcDataAdapter
System.Data.OleDb	OleDbConnection；OleDbCommand；OleDbDataReader；OleDbDataAdapter
System.Data.OracleClient	OracleConnection；OracleCommand；OracleDataReader；OracleDataAdapter

7.2　使用 Connection 连接数据库

本书数据源以 Microsoft SQL Server 2005 数据库为例，也就是说，使用 SQLClient 连接接口，访问数据库使用 SqlConnection、SqlCommand、SqlDataReader 和 SqlDataAdapter 对象。

在建立数据库连接之前，首先在 Web.Config 配置文件中建立一个连接字符串，然后建立数据库连接。

【例 7-1】演示如何建立 Microsoft SQL Server 2005 数据库连接。

(1) 运行 VWD 2008，新建一个名为 Accessdatabase 的 ASP.NET 网站。

(2) 在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【添加新项】命令，在弹出的对话框中选择【SQL Server 数据库】模板，更改名称为 MyDatabase.mdf，创建数据库。如图 7-3 所示。



图 7-3　新建数据库

(3) 单击【添加】按钮，弹出如图 7-4 所示的对话框，单击【确定】按钮，将数据库 MyDatabase.mdf 保存到 App_Data 文件夹中。



图 7-4　创建数据库文件夹

(4) 在【数据库资源管理器】中，双击数据库名 MyDatabase.mdf，数据库资源管理器显示如图 7-5 所示。

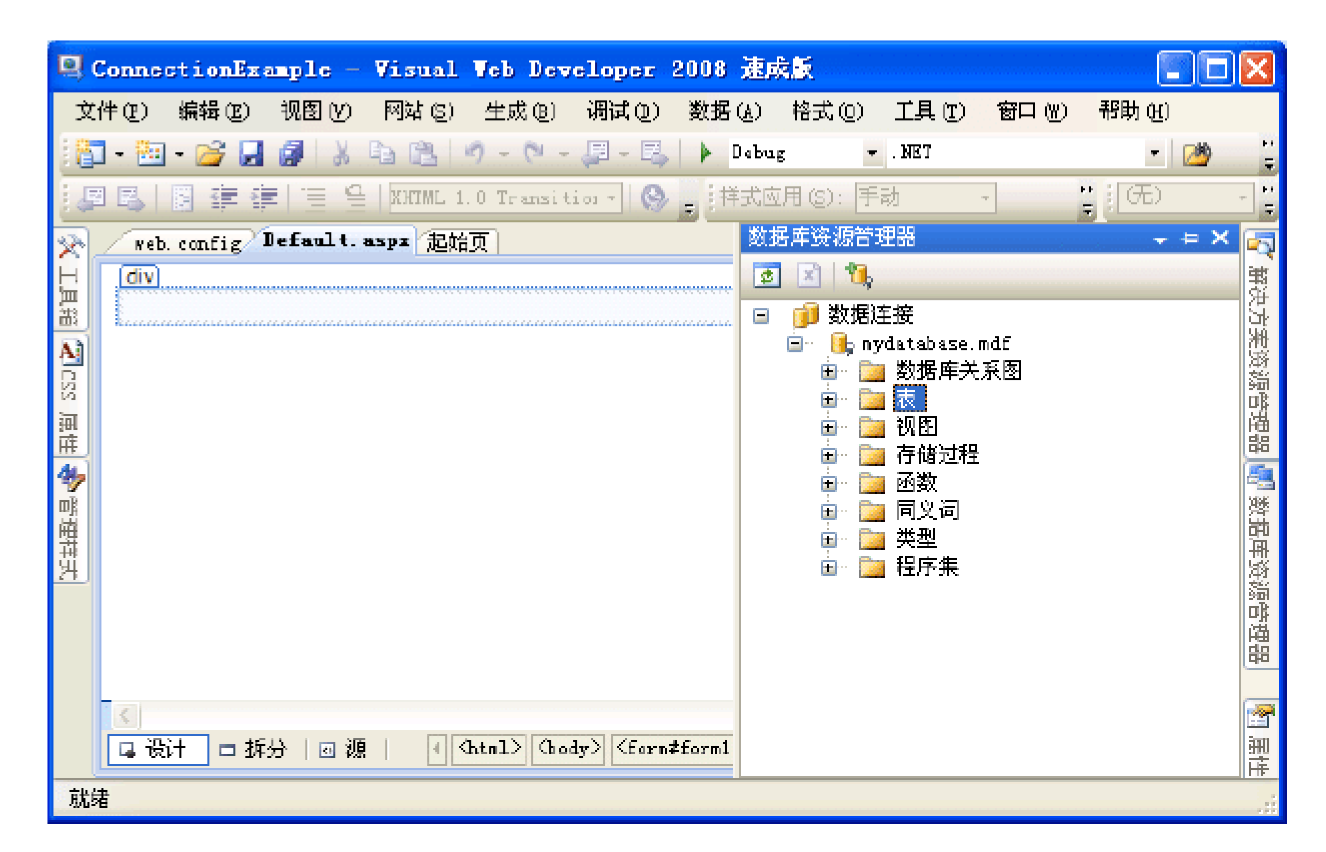


图 7-5　数据库资源管理器

(5) 用鼠标右键单击【表】结点，创建 student 表，表的结构和记录如表 7-2 所示。

表 7-2　student 表的结构和记录

No (varchar，10) (编号，主键)	Name (varchar，50) (姓名)	Sex (char，2) (性别)	birth (datetime) (出生日期)	Address (varchar，50) (地址)	Photo (varchar，50) (照片)
1	张小兵	男	1989-6-29	北京	1.jpg
2	李明	男	1988-5-1	上海	2.jpg
3	王彬	男	1989-9-25	郑州	3.jpg
4	陈飞	男	1988-6-8	洛阳	4.jpg
5	王燕	女	1989-12-11	沈阳	5.jpg
6	陈武	男	1987-7-21	广州	6.jpg
7	刘华	女	1990-3-15	广州	7.jpg
8	翟力	女	1989-6-11	深圳	8.jpg
9	大侠	男	1989-6-1	沈阳	9.jpg

(6) 打开 web.config 配置文件，将<connectionStrings/>标记用下面代码替换。

```
<connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=.\SQLEXPRESS;
     AttachDbFilename=|DataDirectory|\mydatabase.mdf;Integrated Security=True;User Instance=True"/>
</connectionStrings>
```

其中，Data Source 表示 SQL Server 2005 数据库服务器名称，AttachDbFilename 表示数据库的路径和文件名，|DataDirectory| 表示网站默认数据库路径 App_Data。

(7) 在网站中添加一个名为 connection.aspx 的网页，切换到【设计】视图，向该页面拖放一个 Label 控件，使用默认控件名称，然后在 connection.aspx.cs 页面中添加下面代码。

```
//引用数据库访问名称空间
using System.Data.SqlClient;
…
protected void Page_Load(object sender, EventArgs e)
{
  //从 web.config 配置文件取出数据库连接串
  string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
  //建立数据库连接对象
  SqlConnection sqlconn = new SqlConnection(sqlconnstr);
  //打开连接对象
  sqlconn.Open();
  Label1.Text = "成功建立 Sql Server 2005 数据库连接";
  //关闭连接对象
  sqlconn.Close();
  sqlconn = null;
 }
```

(8) 程序运行效果如图 7-6 所示。

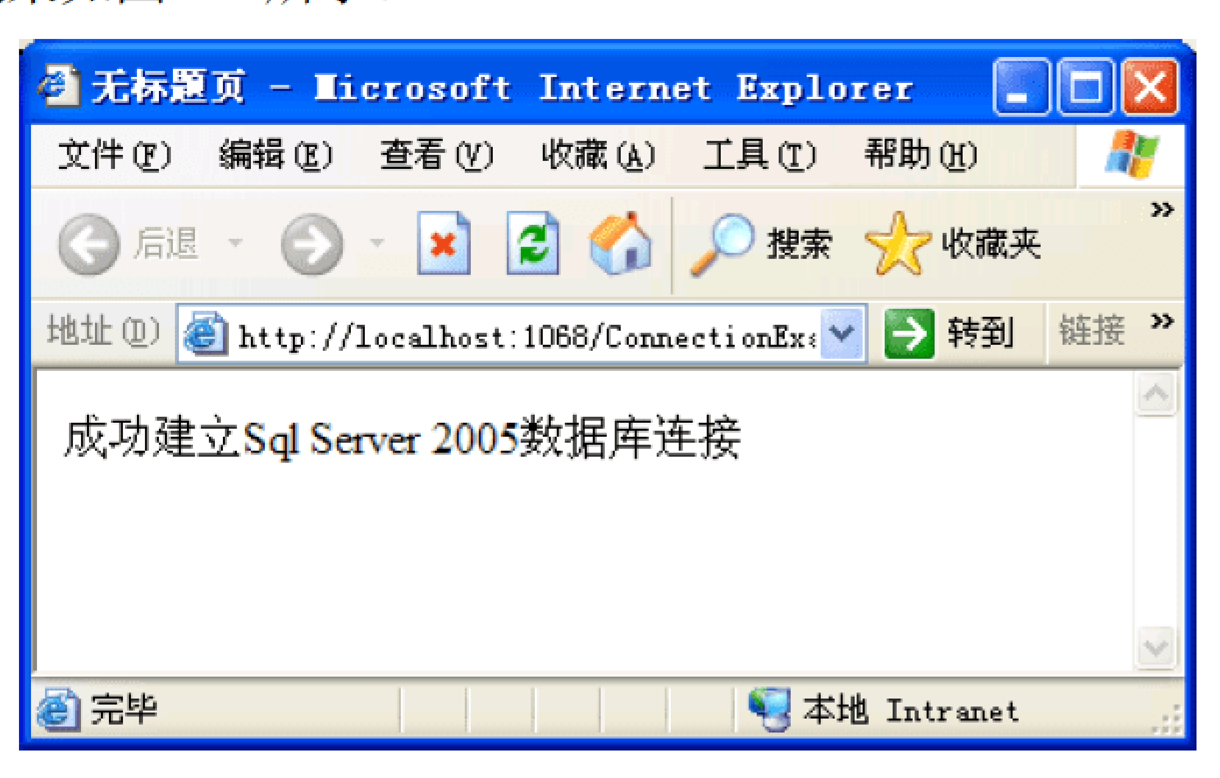


图 7-6　connection.aspx 运行效果

在访问数据库的数据之前，需要使用 Connection 对象的 Open 方法把数据库打开，并在完成数据库的操作之后使用 Connection 对象的 Close 方法将数据库关闭。

7.3　使用 Command 对象执行数据库命令

Command 对象是用来执行数据库操作命令的，比如对数据库中数据表记录的查询、增加、修改或删除等都是要通过 Command 对象来实现的。一个数据库操作命令可以用 SQL 语句来表达，包括 SELECT 语句、UPDATE 语句、DELETE 语句、INSERT 语句等。Command

对象可以传递参数并返回值，同时 Command 也可以调用数据库中的存储过程。

像 Connection 对象一样，对于操作 SQL Server 数据库而言，使用 SqlCommand 对象。Command 对象的属性如表 7-3 所示。

表 7-3　Command 对象的属性

属　　性	说　　明
Connection	包含了数据库连接信息的 Connection 对象
CommandText	要运行的 SQL 命令
CommandType	命令类型
Parameters	Parameters 对象集合

7.3.1　使用 Command 对象查询数据库的数据

使用 Command 对象查询数据库数据的一般步骤为：先建立数据库连接；然后创建 Command 对象，并设置它的 Connection 和 CommandText 两个属性，分别表示数据库连接和需要执行的 SQL 命令；接下来使用 Command 对象的 ExecuteReader 方法，把返回结果放在 DataReader 对象中；最后通过循环，处理数据库查询结果。

【例 7-2】在【例 7-1】基础上，介绍如何使用 Command 对象查询数据库的数据。

(1) 在 Accessdatabase 网站中添加一个名为 command_select.aspx 的网页，切换到【设计】视图，向该页面拖放一个 Label 控件，使用默认控件名称。

(2) 在 command_select.aspx.cs 页面中添加下面代码。

```
//引用数据库访问名称空间
using System.Data.SqlClient;
  •••
protected void Page_Load(object sender, EventArgs e)
{
   string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
   SqlConnection sqlconn = new SqlConnection(sqlconnstr);
   //建立 Command 对象
   SqlCommand sqlcommand = new SqlCommand();
   //给 sqlcommand 的 Connection 属性赋值
   sqlcommand.Connection = sqlconn;
   //打开连接
   sqlconn.Open();
   //SQL 命令赋值
   sqlcommand.CommandText = "select * from student";
   //建立 DataReader 对象，并返回查询结果
   SqlDataReader sqldatareader=sqlcommand.ExecuteReader();
   //逐行遍历查询结果
   while(sqldatareader.Read())
```

```
    {
        Label1.Text += sqldatareader.GetString(0) + " ";
        Label1.Text += sqldatareader.GetString(1)   + " ";
        Label1.Text += sqldatareader.GetString(2)   + " ";
        Label1.Text += sqldatareader.GetDateTime(3)   + " ";
        Label1.Text += sqldatareader.GetString(4)   + " ";
        Label1.Text += sqldatareader.GetString(5)   + "<br />";
    };
    sqlcommand = null;
    sqlconn.Close();
    sqlconn = null;
}
```

(3) 程序运行效果如图 7-7 所示。

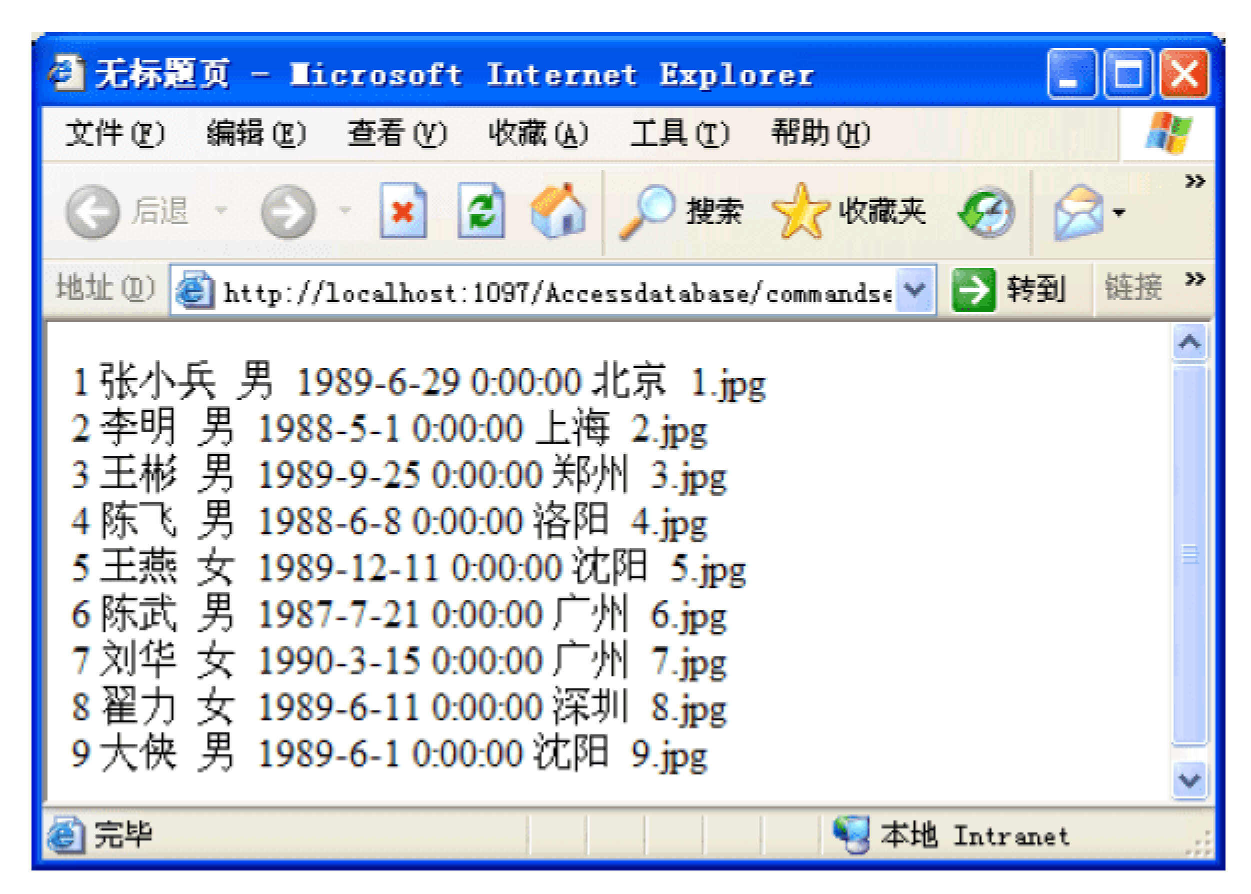


图 7-7　command_select.aspx 运行效果

7.3.2　使用 Command 对象增加数据库的数据

使用 Command 对象增加数据库数据的一般步骤为：先建立数据库连接；然后创建 Command 对象，设置它的 Connection 和 CommandText 两个属性，并使用 Command 对象的 Parameters 属性来设置输入参数；最后使用 Command 对象的 ExecuteNonquery 方法执行数据库数据增加命令，其中 ExecuteNonquery 方法表示要执行的是没有返回数据的命令。

【例 7-3】演示如何使用 Command 对象增加新数据。

(1) 在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【新建文件夹】命令，新建文件夹，改名为 image，用于存放学生照片。

(2) 在 Accessdatabase 网站中添加一个名为 command_insert.aspx 的网页。

(3) 将 command_insert 页面设计为如图 7-8 所示的形式。

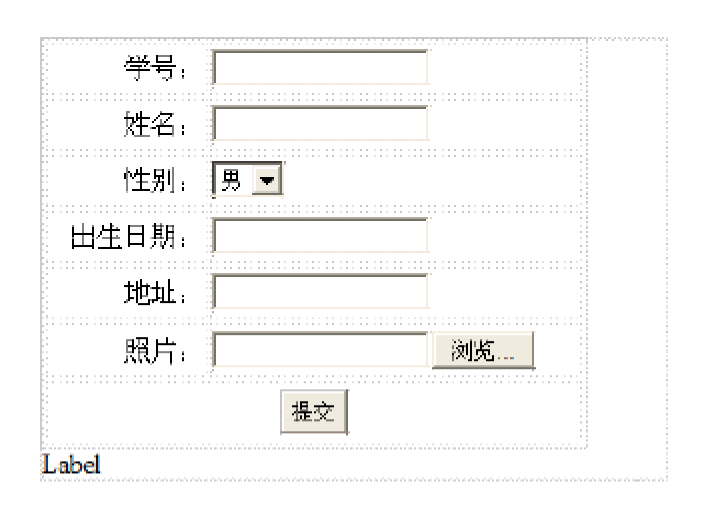


图 7-8　command_insert.aspx 的设计页面

对应【源】视图中的代码如下：

```
<table style="width: 320px; height: 240px">
        <tr>
        <td style="width: 100px; text-align: right"> 学号：</td>
        <td style="width: 220px">
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>   </tr>
        <tr>
        <td style="width: 100px; text-align: right"> 姓名：</td>
        <td style="width: 220px">
 <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>   </tr>
        <tr>
        <td style="width: 100px; text-align: right"> 性别：</td>
        <td style="width: 220px">
          <asp:DropDownList ID="DropDownList1" runat="server">
                <asp:ListItem Selected="True">男</asp:ListItem>
                <asp:ListItem>女</asp:ListItem>
          </asp:DropDownList>   </td>    </tr>
        <tr>
        <td style="width: 100px; text-align: right">  年龄：</td>
        <td style="width: 220px">
               <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox></td> </tr>
        <tr>
        <td style="width: 100px; text-align: right"> 地址：</td>
        <td style="width: 220px">
               <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox></td>   </tr>
        <tr>
        <td style="width: 100px; text-align: right"> 照片：</td>
        <td style="width: 220px">
               <asp:FileUpload ID="FileUpload1" runat="server" /></td> </tr>
        <tr>
        <td colspan="2" style="text-align: center">
         <asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click" /></td> </tr>
    </table>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

(4) 双击设计视图中的【提交】按钮，添加如下所示的后台代码。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //建立 Command 对象
    SqlCommand sqlcommand = new SqlCommand();
    sqlcommand.Connection = sqlconn;
    //把 SQL 语句赋给 Command 对象
    sqlcommand.CommandText = "insert into student(no,name,sex,birth,address,photo)
        values (@no,@name,@sex,@birth,@address,@photo)";
    sqlcommand.Parameters.AddWithValue("@no",TextBox1.Text);
    sqlcommand.Parameters.AddWithValue("@name",TextBox2.Text);
    sqlcommand.Parameters.AddWithValue("@sex",DropDownList1.Text);
    sqlcommand.Parameters.AddWithValue("@birth",TextBox3.Text);
    sqlcommand.Parameters.AddWithValue("@address",TextBox4.Text);
    sqlcommand.Parameters.AddWithValue("@photo",FileUpload1.FileName);
    try
    {
        //打开连接
        sqlconn.Open();
        //执行 SQL 命令
        sqlcommand.ExecuteNonQuery();
        //把学生的照片上传到网站的 image 文件夹中
        if (FileUpload1.HasFile == true)
        {
            FileUpload1.SaveAs(Server.MapPath(("~/image/") + FileUpload1.FileName));
        }
        Label1.Text = "成功增加记录";
    }
    catch (Exception ex)
    {
        Label1.Text = "错误原因：" + ex.Message;
    }
    finally
    {
        sqlcommand = null;
        sqlconn.Close();
        sqlconn = null;
    }
}
```

(5) 程序运行效果如图 7-9 所示。

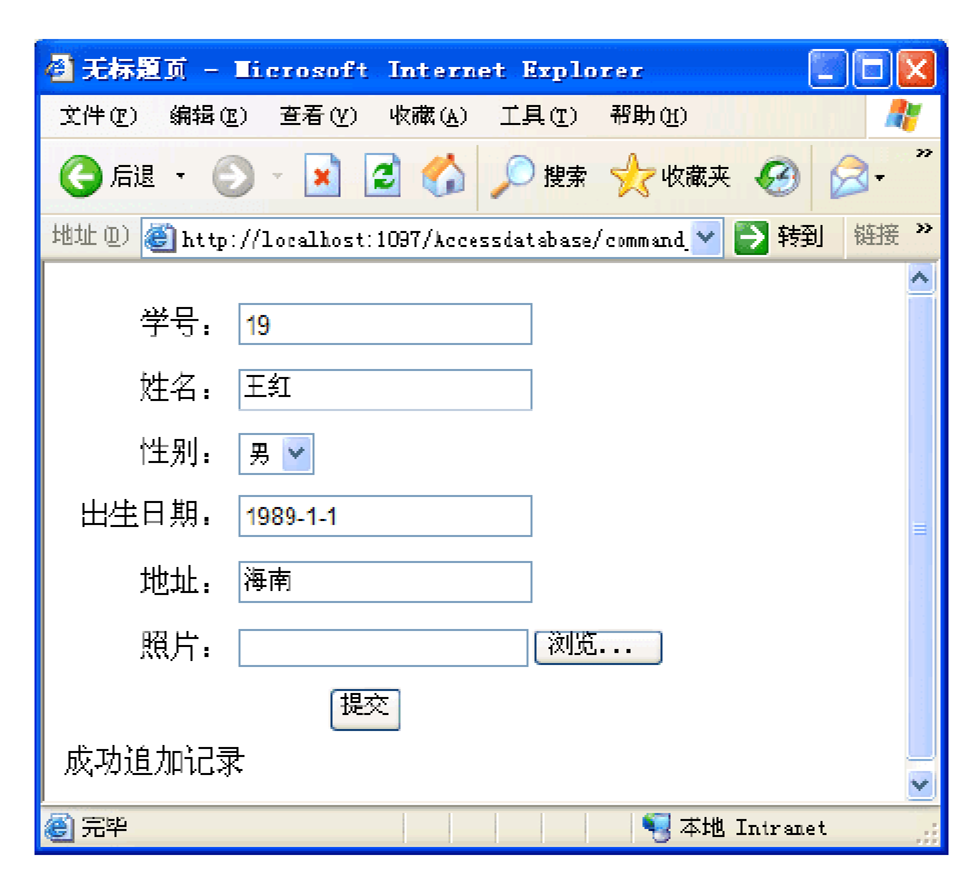


图 7-9　command_insert.aspx 的运行效果

7.3.3　使用 Command 对象删除数据库的数据

使用 Command 对象删除数据库数据的一般步骤为：先建立数据库连接；然后创建 Command 对象，设置它的 Connection 和 CommandText 两个属性，并使用 Command 对象的 Parameters 属性来传递参数；最后使用 Command 对象的 ExecuteNonquery 方法执行数据库数据删除命令。

【例 7-4】使用 Command 对象删除数据。

(1) 在 Accessdatabase 网站中添加一个名为 command_delete.aspx 的网页。

(2) 向 command_delete 页面添加 2 个 Label 控件，1 个 TextBox 控件，1 个 Button 控件，其中 Button 控件作为【删除】按钮。

(3) 双击设计视图中的【删除】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
  •••
protected void Button1_Click(object sender, EventArgs e)
{
  int intDeleteCount;
  string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
  SqlConnection sqlconn = new SqlConnection(sqlconnstr);
  //建立 Command 对象
  SqlCommand sqlcommand = new SqlCommand();
  //给 Command 对象的 Connection 和 CommandText 属性赋值
  sqlcommand.Connection = sqlconn;
  sqlcommand.CommandText = "delete from student where no=@no";
  sqlcommand.Parameters.AddWithValue("@no",TextBox1.Text);
  try
  {
     sqlconn.Open();
     intDeleteCount=sqlcommand.ExecuteNonQuery();
```

```
            if (intDeleteCount>0)
                Label1.Text = "成功删除记录";
            else
                Label1.Text = "该记录不存在";
        }
        catch (Exception ex)
        {
            Label1.Text = "错误原因："+ex.Message;
        }
        finally
        {
            sqlcommand = null;
            sqlconn.Close();
            sqlconn = null;
        }
    }
```

(4) 程序运行效果如图 7-10 所示。



图 7-10　command_delete.aspx 的运行效果

7.3.4　使用 Command 对象修改数据库的数据

使用 Command 对象修改数据库数据的一般步骤为：先建立数据库连接；然后创建 Command 对象，设置它的 Connection 和 CommandText 两个属性，并使用 Command 对象的 Parameters 属性来传递参数；接下来使用 Command 对象的 ExecuteNonquery 方法执行数据库数据修改命令。【例 7-5】同时说明存储过程的调用方法。

【例 7-5】演示如何使用 Command 对象修改数据。

(1) 打开 Accessdatabase 网站，为 MyDatabase.mdf 数据库建立名为 update_student 的存储过程。在【数据库资源管理器】中，用鼠标右键单击【存储过程】结点，如图 7-11 所示。



图 7-11　数据库资源管理器

选择【添加新存储过程】命令，在存储过程定义窗口中，添加如下代码，最后单击工具栏中的【保存】按钮，保存存储过程到数据库中。

```
CREATE  PROCEDURE dbo.update_student
(
    //入口参数
    @no varchar(10),
    @name varchar(50),
    @sex varchar(2) ,
    @birth   datetime,
    @address varchar(50),
    @photo varchar(50)
)
AS
    //修改学号为@no 的学生信息
    update student set name=@name,sex=@sex,birth=@birth,address=@address,
                photo=@photo where no=@no
    RETURN
```

(2) 在 Accessdatabase 网站中添加一个名为 command_update.aspx 的网页，将 command_update 页面设计为如图 7-8 所示的形式。

(3) 双击设计视图中的【提交】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
  •••
protected void Button1_Click(object sender, EventArgs e)
{
    int intUpdateCount;
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //建立 Command 对象
    SqlCommand sqlcommand = new SqlCommand();
    sqlcommand.Connection = sqlconn;
```

```
        //把存储过程名称赋给 Command 对象的 CommandText 属性
        sqlcommand.CommandText = "update_student";
        //说明命令类型为存储过程
        sqlcommand.CommandType = CommandType.StoredProcedure;
        sqlcommand.Parameters.AddWithValue("@no", TextBox1.Text);
        sqlcommand.Parameters.AddWithValue("@name", TextBox2.Text);
        sqlcommand.Parameters.AddWithValue("@sex", DropDownList1.Text);
        sqlcommand.Parameters.AddWithValue("@birth", TextBox3.Text);
        sqlcommand.Parameters.AddWithValue("@address", TextBox4.Text);
        sqlcommand.Parameters.AddWithValue("@photo", FileUpload1.FileName);
        try
        {
            //打开连接
            sqlconn.Open();
            //执行 SQL 命令
            intUpdateCount=sqlcommand.ExecuteNonQuery();
            //把学生的照片上传到网站的 image 文件夹中
            if (FileUpload1.HasFile == true)
            {
                FileUpload1.SaveAs(Server.MapPath(("~/image/") + FileUpload1.FileName));
            }
            if (intUpdateCount > 0)
                Label1.Text = "成功修改记录";
            else
                Label1.Text = "该记录不存在";
        }
        catch (Exception ex)
        {
            Label1.Text = "错误原因：" + ex.Message;
        }
        finally
        {
            sqlcommand = null;
            sqlconn.Close();
            sqlconn = null;
        }
    }
```

(4) 程序运行效果如图 7-12 所示。

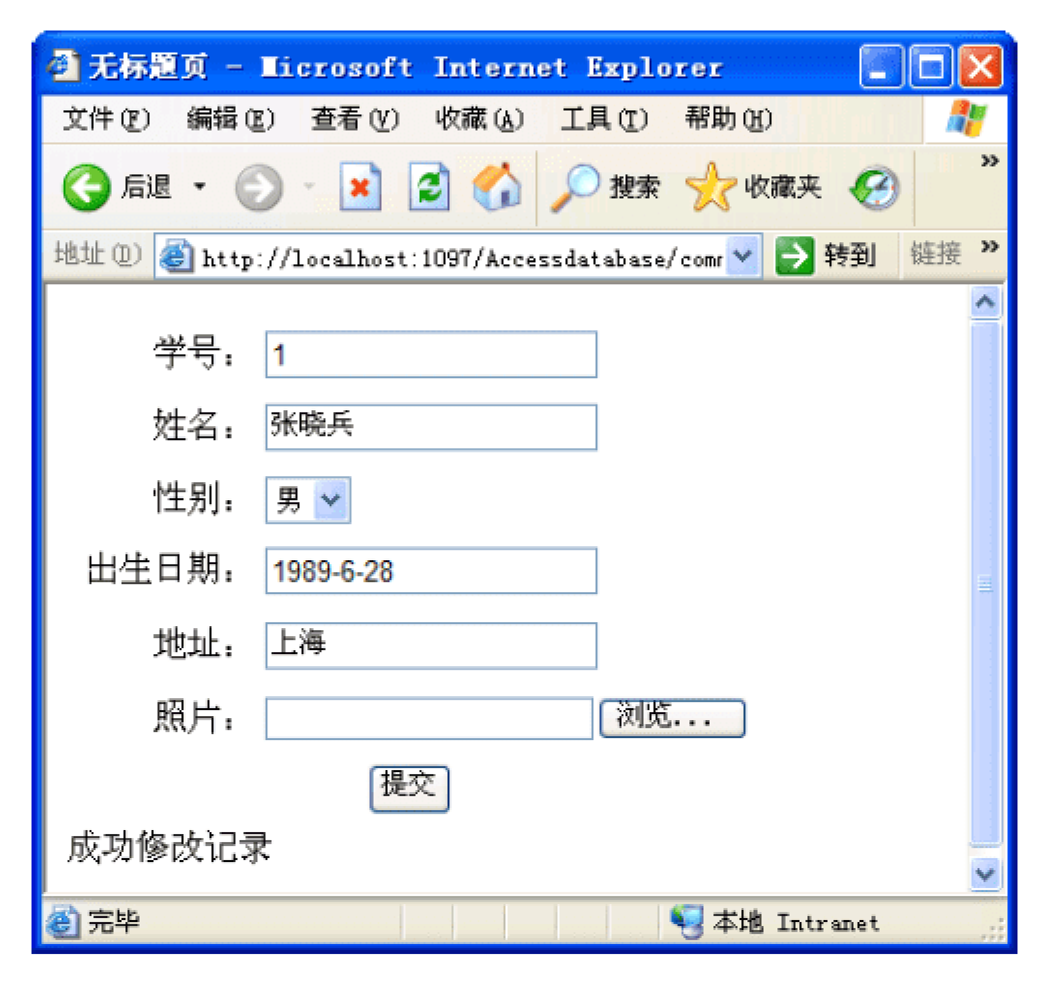


图 7-12　command_update.aspx 的运行效果

7.3.5　数据库事务处理

对于数据库管理系统来说，如果没有显式定义事务的开始和结束，就默认一条 SQL 语句为一个单独事务，多数情况下采用这种默认方式就足够了。但是，有时需要将一组 SQL 语句作为一个事务，要不全做，要不全不做。

在 ASP.NET 中，可以使用 Connection 和 Transaction 对象开始、提交和回滚事务。一般步骤为：调用 Connection 对象的 BeginTransaction 方法来标记事务的开始，BeginTransaction 方法返回对 Transaction 的引用；将 Transaction 对象赋给 Command 的 Transaction 属性；执行事务操作；如果事务操作成功，使用 Transaction 对象的 Commit 方法提交事务，否则，使用 Rollback 方法回滚事务。

【例 7-6】演示事务处理。

(1) 在 Accessdatabase 网站中添加一个名为 transaction.aspx 的网页。

(2) 向 transaction 页面添加 1 个 Label 控件，1 个 Button 控件，其中 Button 控件作为【事务提交】按钮。

(3) 双击设计视图中的【事务提交】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
  •••
protected void Button1_Click(object sender, EventArgs e)
{
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    sqlconn.Open();
    //开始事务
    SqlTransaction tran=sqlconn.BeginTransaction();
    SqlCommand sqlcommand = new SqlCommand();
    sqlcommand.Connection = sqlconn;
    sqlcommand.Transaction = tran;
```

```
        try
        {
            sqlcommand.CommandText = "update student set address='beijing' where no=1";
            sqlcommand.ExecuteNonQuery();
            sqlcommand.CommandText = "update student set address='zhengzhou' where no=2";
            sqlcommand.ExecuteNonQuery();
            tran.Commit();
            Label1.Text = "事务提交成功";
        }
        catch (Exception ex)
        {
            tran.Rollback();
            Label1.Text = "事务提交失败：" + ex.Message;
        }
        finally
        {
            sqlcommand = null;
            sqlconn.Close();
            sqlconn = null;
        }
    }
```

从 7.3 节例子中可以看出，Command 对象的 ExecuteNonQuery 方法在执行数据更新 SQL 语句(如 INSERT，UPDATE 或 DELETE)时使用，这些语句的共同特点是没有返回数据。此外，ExecuteNonQuery 方法可以返回一个整数，表示已经执行语句在数据库中影响数据的行数。

如果需要执行有返回结果的 SQL 语句(如 SELECT)，那么就需要使用 Command 对象的 ExecuteReader 方法，并将执行结果放到 DataReader 中。这个对象是一个专门读取数据的对象，除了能做读取数据工作之外，其他什么也不能做，所以这是一种简单的读取数据的方法。

7.4　使用 DataAdapter 对象执行数据库命令

对于 SQL SERVER 接口，使用的是 SqlDataAdapter 对象，在使用 DataAdapter 对象时，只需分别设置表示 SQL 命令和数据库连接的两个参数，就可以通过它的 Fill 方法把查询结果放在一个 DataSet 对象中。

在一个 DataSet 对象实例中，可以包含多个 DataTable，而一个 DataTable 可以包含多个 DataRow。

当把一个 DataSet 中的一个数据表复制到一个 DataTable 中之后，可以通过对 DataTable 数据的访问来实现对 DataSet 中数据的访问了。除此之外，还可以通过修改 DataTable 中的数据来更新 DataSet。

DataRow 表示 DataTable 的数据行，一个 DataTable 中的数据行会有很多。针对一个 DataTable，它的 Rows 属性表示这个表的所有数据行，是一个集合，类名为 DataRowCollection，它的每个元素的类型是 DataRow。

通过 7.4 节的例子说明使用 DataAdapter 和 DataSet 读取和修改数据库数据的方法。

7.4.1　使用 DataAdapter 对象查询数据库的数据

使用 DataAdapter 对象查询数据库数据的一般步骤为：首先建立数据库连接；然后利用数据库连接和 SELECT 语句建立 DataAdapter 对象，并使用 DataAdapter 对象的 Fill 方法把查询结果放在 DataSet 对象的一个数据表中；接下来将该数据表复制到 DataTable 对象中；最后实现对 DataTable 对象中数据的查询。

【例 7-7】演示如何使用 DataAdapter 对象查询数据库的数据。

(1) 在 Accessdatabase 网站中添加一个名为 DataAdapter_select.aspx 的网页，切换到【设计】视图，向该页面拖放一个 Label 控件，使用默认控件名称。

(2) 在 DataAdapter _select.aspx.cs 页面中添加如下所示的代码。

```
//引用数据库访问名称空间
using System.Data.SqlClient;
  •••
protected void Page_Load(object sender, EventArgs e)
{
  string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
  SqlConnection sqlconn = new SqlConnection(sqlconnstr);
  //建立 DataSet 对象
  DataSet ds = new DataSet();
  //建立 DataTable 对象
  DataTable dtable;
  //建立 DataRowCollection 对象
  DataRowCollection coldrow;
  //建立 DataRow 对象
  DataRow drow;
  //打开连接
  sqlconn.Open();
  //建立 DataAdapter 对象
  SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
  //用 Fill 方法返回的数据，填充 DataSet，数据表取名为 tabstudent
  sqld.Fill(ds, "tabstudent");
  //将数据表 tabstudent 的数据复制到 DataTable 对象
  dtable = ds.Tables["tabstudent"];
  //用 DataRowCollection 对象获取这个数据表的所有数据行
  coldrow = dtable.Rows;
  //逐行遍历，取出各行的数据
```

```
        for (int inti = 0; inti < coldrow.Count; inti++)
        {
            drow = coldrow[inti];
            Label1.Text += "学号：" + drow[0];
            Label1.Text += "  姓名：" + drow[1];
            Label1.Text += "  性别：" + drow[2];
            Label1.Text += "  出生日期：" + drow[3];
            Label1.Text += "  地址：" + drow[4] + "<br />";
        }
        sqlconn.Close();
        sqlconn = null;
    }
```

(3) 程序运行效果如图 7-13 所示。

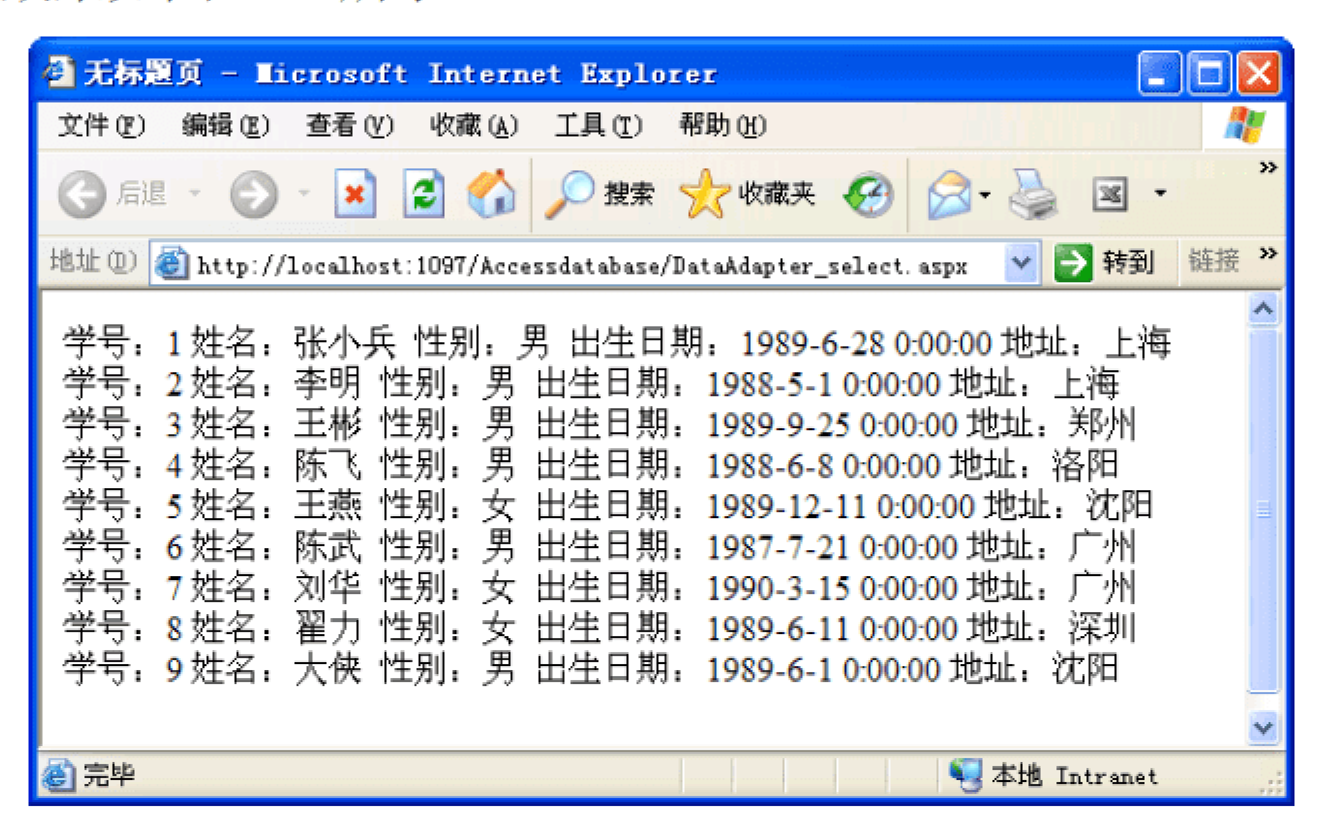


图 7-13　DataAdapter _select.aspx 运行效果

关于显示 DataSet 中的数据还有更简单的方法，就是绑定 GridView 控件等，详细内容在第 8 章中进行介绍。

7.4.2　使用 DataAdapter 对象修改数据库的数据

使用 DataAdapter 对象修改数据库数据的一般步骤为：首先建立数据库连接；然后利用数据库连接和 SELECT 语句建立 DataAdapter 对象；并配置它的 UpdateCommand 属性，定义修改数据库的 UPDATE 语句；使用 DataAdapter 对象的 Fill 方法把 SELECT 语句的查询结果放在 DataSet 对象的数据表中；接下来将该数据表复制到 DataTable 对象中；最后实现对 DataTable 对象中数据的修改，并通过 DataAdapter 对象的 Update 方法向数据库提交修改数据。

【例 7-8】演示如何使用 DataAdapter 对象修改数据库的数据。

(1) 在 Accessdatabase 网站中添加一个名为 DataAdapter_update.aspx 的网页。

(2) 向 DataAdapter_ update 页面添加 1 个 Label 控件，1 个 Button 控件，其中 Button 控件作为【更新】提交按钮。

(3) 双击设计视图中的【更新】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
   •••
protected void Button1_Click(object sender, EventArgs e)
{
   string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
   SqlConnection sqlconn = new SqlConnection(sqlconnstr);
   //建立 DataSet 对象
   DataSet ds = new DataSet();
   //建立 DataTable 对象
   DataTable dtable;
   //建立 DataRowCollection 对象
   DataRowCollection coldrow;
   //建立 DataRow 对象
   DataRow drow;
   //打开连接
   sqlconn.Open();
   //建立 DataAdapter 对象
   SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
   //自己定义 Update 命令，其中@NAME，@NO 是两个参数
   sqld.UpdateCommand = new SqlCommand("UPDATE student SET NAME = @NAME
            WHERE NO = @NO", sqlconn);
   //定义@NAME 参数，对应于 student 表的 NAME 列
   sqld.UpdateCommand.Parameters.Add("@NAME", SqlDbType.VarChar, 50, "NAME");
   //定义@NO 参数，对应于 student 表的 NO 列，而且@NO 是修改前的原值
   SqlParameter parameter = sqld.UpdateCommand.Parameters.Add("@NO", SqlDbType.VarChar, 10);
   parameter.SourceColumn = "NO";
   parameter.SourceVersion = DataRowVersion.Original;
   //用 Fill 方法返回的数据，填充 DataSet，数据表取名为 tabstudent
   sqld.Fill(ds, "tabstudent");
   //将数据表 tabstudent 的数据复制到 DataTable 对象
   dtable = ds.Tables["tabstudent"];
   //用 DataRowCollection 对象获取这个数据表的所有数据行
   coldrow = dtable.Rows;
   //修改操作，逐行遍历，取出各行的数据
   for (int inti = 0; inti < coldrow.Count; inti++)
   {
       drow = coldrow[inti];
       //给每位学生姓名后加上字母 A
       drow[1]=drow[1]+"A";
   }
   //提交更新
   sqld.Update(ds, "tabstudent");
   sqlconn.Close();
   sqlconn = null;
```

```
    Label1.Text = "更新成功";
}
```

(4) 程序运行效果如图 7-14 所示。

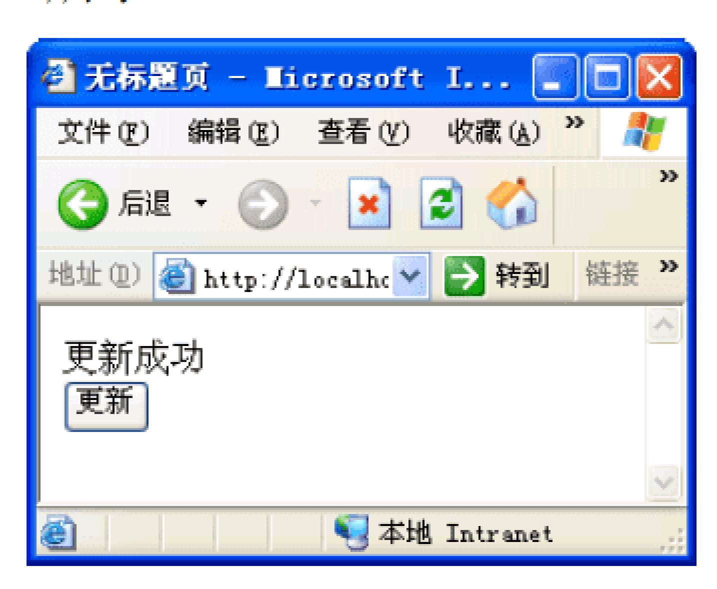


图 7-14　DataAdapter_ update.aspx 的运行效果

7.4.3　使用 DataAdapter 对象增加数据库的数据

使用 DataAdapter 对象增加数据库数据的一般步骤为：首先建立数据库连接；然后利用数据库连接和 SELECT 语句建立 DataAdapter 对象；并建立 CommandBuilder 对象自动生成 DataAdapter 的 Command 命令，否则就要自己给 UpdateCommand、InsertCommand、DeleteCommand 属性定义 SQL 更新语句；使用 DataAdapter 对象的 Fill 方法把 SELECT 语句的查询结果放在 DataSet 对象的数据表中；接下来将该数据表复制到 DataTable 对象中；最后实现对 DataTable 对象中数据的增加，并通过 DataAdapter 对象的 Update 方法向数据库提交数据。

【例 7-9】演示如何使用 DataAdapter 对象增加一条学生记录。

(1) 在 Accessdatabase 网站中添加一个名为 DataAdapter_insert.aspx 的网页。

(2) 向 DataAdapter_insert 页面添加 1 个 Label 控件，1 个 Button 控件，其中 Button 控件作为【增加】提交按钮。

(3) 双击设计视图中的【增加】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
  •••
protected void Button1_Click(object sender, EventArgs e)
{
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //建立 DataSet 对象
    DataSet ds = new DataSet();
    //建立 DataTable 对象
    DataTable dtable;
    //建立 DataRow 对象
    DataRow drow;
    //打开连接
```

```
    sqlconn.Open();
    //建立 DataAdapter 对象
    SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
    //建立 CommandBuilder 对象来自动生成 DataAdapter 的 Command 命令，否则就要自己编写
    //Insertcommand ,deletecommand , updatecommand 命令。
    SqlCommandBuilder cb = new SqlCommandBuilder(sqld);
    //用 Fill 方法返回的数据，填充 DataSet，数据表取名为 tabstudent
    sqld.Fill(ds, "tabstudent");
    //将数据表 tabstudent 的数据复制到 DataTable 对象
    dtable = ds.Tables["tabstudent"];
    //增加新记录
    drow = ds.Tables["tabstudent"].NewRow();
    //给该记录赋值
    drow[0] = "19";
    drow[1] = "陈峰";
    drow[2] = "男";
    ds.Tables["tabstudent"].Rows.Add(drow);
    //提交更新
    sqld.Update(ds, "tabstudent");
    sqlconn.Close();
    sqlconn = null;
    Label1.Text = "增加成功";
}
```

(4) 程序运行效果如图 7-15 所示。



图 7-15　DataAdapter_insert.aspx 的运行效果

7.4.4　使用 DataAdapter 对象删除数据库的数据

使用 DataAdapter 对象删除数据库数据的一般步骤为：首先建立数据库连接；然后利用数据库连接和 SELECT 语句建立 DataAdapter 对象；并建立 CommandBuilder 对象自动生成 DataAdapter 的 Command 命令；使用 DataAdapter 对象的 Fill 方法把 SELECT 语句的查询结果放在 DataSet 对象的数据表中；接下来将该数据表复制到 DataTable 对象中；最后实现对 DataTable 对象中数据的删除，并通过 DataAdapter 对象的 Update 方法向数据库提交数据。

【例 7-10】演示如何使用 DataAdapter 对象删除符合条件的学生记录。

(1) 在 Accessdatabase 网站中添加一个名为 DataAdapter_delete.aspx 的网页。

(2) 向 DataAdapter_delete 页面添加 1 个 Label 控件，1 个 Button 控件，其中 Button 控件作为【删除】按钮。

(3) 双击设计视图中的【删除】按钮，添加如下所示的后台代码。

```
using System.Data.SqlClient;
  ...
protected void Button1_Click(object sender, EventArgs e)
{
   string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
   SqlConnection sqlconn = new SqlConnection(sqlconnstr);
   DataSet ds = new DataSet();
   DataTable dtable;
   DataRowCollection coldrow;
   DataRow drow;
   sqlconn.Open();
   //建立 DataAdapter 对象
   SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
   //建立 CommandBuilder 对象来自动生成 DataAdapter 的 Command 命令，否则就要自己编写
   //Insertcommand ,deletecommand , updatecommand 命令。
    SqlCommandBuilder cb = new SqlCommandBuilder(sqld);
    //用 Fill 方法返回的数据，填充 DataSet，数据表取名为 tabstudent
    sqld.Fill(ds, "tabstudent");
    dtable = ds.Tables["tabstudent"];
    coldrow = dtable.Rows;
    //逐行遍历，删除地址为空的记录
    for (int inti = 0; inti < coldrow.Count; inti++)
    {
         drow = coldrow[inti];
         if (drow["address"].ToString()== "")
              drow.Delete();
    }
    //提交更新
    sqld.Update(ds, "tabstudent");
    sqlconn.Close();
    sqlconn = null;
    Label1.Text = "删除成功";
}
```

(4) 程序运行效果如图 7-16 所示。



图 7-16　DataAdapter_delete.aspx 的运行效果

实验 7　ADO.NET 数据库访问

1. 实验目的

熟悉 ADO.NET 数据库访问技术，掌握 Command 和 DataAdapter 对象操作数据库数据的方法。

2. 实验内容和要求

(1) 新建名字为 Accessdatabase_ Exercise 的网站。

(2) 在网站的 App_Data 文件夹中，建立数据库 MyDatabase_ Exercise.mdf。

(3) 在该数据库中建立一张职工表，并且添加一些模拟的职工记录。其关系模式如下：

Employees(ID，NAME，SEX，AGE，Dateofwork，FilenameofPhoto)

(4) 在 web.config 配置文件中，修改<connectionStrings/>标记如下。

```
<connectionStrings>
 <add name="ConnectionString" connectionString="Data Source=.\SQLEXPRESS;
      AttachDbFilename=|DataDirectory|\ MyDatabase_ Exercise.mdf;Integrated Security=True;User
      Instance=True"/>
</connectionStrings>
```

(5) 添加一个网页，利用 Command 对象实现新职工的录入。

(6) 添加一个网页，利用 Command 对象实现删除指定编号的职工记录。

(7) 添加一个网页，利用 Command 对象实现修改指定编号的职工信息。

(8) 添加一个网页，利用 DataAdapter 对象实现查询职工信息，并显示到网页的 Label 控件上。

第8章　ASP.NET中的数据绑定

本章首先介绍单值绑定和列表控件的数据绑定过程，然后介绍 GridView 等复杂数据绑定控件的基本用法，主要涉及以下 3 种复杂数据绑定控件：GridView、DataList 和 FormView。本章将通过一系列实例逐步学习如何控制控件的显示效果和行为。

本章学习目标

- 掌握单值和列表控件的数据绑定
- 掌握 GridView 的数据绑定方式、绑定列模板列的配置和用法以及数据的排序和分页。
- 掌握 DataList 和 FormView 的数据绑定和自定义样式的设置。

8.1　数据绑定概述

第 7 章学习了如何通过 ADO.NET 访问数据库，而这一章将着重学习如何利用 ASP.NET 提供的控件将数据呈现在页面上。大家知道，Web 系统的一个典型的特征是后台对数据的访问和处理与前台数据的显示分离，而前台显示是通过 HTML 来实现的。一种将数据呈现的最直接的方式是将需要显示的数据和 HTML 标记拼接成字符串并输出，但这种方案的缺点也是显而易见的，不但复杂而且难以重用，尤其是有大宗数据需要处理时。因此为了简化开发过程，ASP.NET 环境中提供了多种不同的服务器端控件来帮助程序员更快速高效地完成数据的呈现。这些用于数据呈现的 ASP.NET 控件，集成了常见的数据显示框架和数据处理功能，因而在使用时只需要设置某些属性，并将需要显示的数据交付给控件，控件就可以帮助用户按照固定的样式(例如表格)或通过模板自定义样式将一系列数据呈现出来，并自动继承某些内置的数据处理功能，例如：排序、分页等，当然也可以通过编程定制或扩展控件的行为。这些控件就被称为数据绑定控件，而将数据交付给数据绑定控件的过程就被称为数据绑定。

数据绑定控件本质上依然是通过 HTML 来呈现数据的，只不过按照某种样式生成 HTML 框架并将数据填入其中的工作由控件自动完成了，一些复杂的数据绑定控件还提供了大量的功能帮助用户对数据进行进一步操作，例如：排序、过滤、新增、修改和删除等等，因而使得数据呈现的过程变得简单而灵活。正因为如此，数据绑定控件的使用是 ASP.NET 编程中非常重要的一部分内容。

本节首先从简单的单值控件和列表控件讲起，然后通过多个例子详细地介绍一种最常用

的数据绑定控件 GridView 的各个方面，最后介绍 DataList、FormView 和 Repeater 这 3 种用法相近的数据绑定控件。

8.2　单值和列表控件的数据绑定

数据绑定本质上是将数据在前台页面上呈现，ASP.NET 3.5 的页面元素可以简单地分成两类：HTML 标记和服务器控件，因此数据绑定实际上是在 HTML 标记中或服务器控件中设置要显示数据的过程。对于页面中的 HTML 标记，可以直接嵌入数据或绑定表达式来设置要显示的数据，而对于服务器控件来说，通常通过设置控件属性或指定数据源来完成数据的绑定，并控制其呈现的样式。常用的绑定表达式具有以下形式：<%#XXX%>，绑定表达式可以直接嵌入到前台页面代码中去，通常用于 HTML 标记中的数据显示或单值控件数据设置，例如：Label、TextBox 等。而对于列表控件(如：DropDownList、CheckBoxList)，以及后面要着重介绍的复杂数据绑定控件则常采用设置数据源的方式完成数据呈现。

8.2.1　单值绑定

【例 8-1】演示了单值绑定的方法，这里将通过绑定表达式和后台设置属性两种方式设置在 HTML 标记中和一个简单服务器控件中要显示的数据，对单值控件的数据设置方式是通用的，因此这里只以 TextBox 为例。

【例 8-1】演示如何进行单值绑定。

(1) 运行 VWD 2008，新建一个名为 DataBinding 的 ASP.NET 网站，并在网站中添加一个名为 SingleValueBinding.aspx 的网页。

(2) 在<div/>标记中添加 2 个 TextBox 控件，并修改页面代码如下：

```
<div>
     <%# SingleValueBindingStr + "1" %>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:TextBox ID="TextBox2" runat="server"
    Text="<%# SingleValueBindingStr + 3 %>"></asp:TextBox>
</div>
```

(3) 修改页面后台代码如下：

```
public partial class SingleValueBinding : System.Web.UI.Page
{
    //在页面代码中将通过绑定表达式直接引用该成员
    public String SingleValueBindingStr = "单值绑定";
        protected void Page_Load(object sender, EventArgs e)
    {
        //页面的数据绑定方法，对于绑定表达式来说是关键的一步
        Page.DataBind();
```

```
            //通过在后台设置服务器控件属性来绑定数据
            this.TextBox1.Text = this.SingleValueBindingStr + "2";
        }
    }
```

(4) 程序运行效果如图 8-1 所示。



图 8-1　SingleValueBinding.aspx 运行效果

在【例 8-1】中，首先用绑定表达式<%# SingleValueBindingStr + "1" %>直接嵌入到 div 标记中来设置 HTML 标记中的显示值，然后通过绑定表达式和后台设置控件属性两种方式绑定了 TextBox 控件的显示数据。

这里需要注意绑定表达式的用法，在<%#%>标记中通过直接引用页面类中定义的公有数据成员 SingleValueBindingStr 构成表达式，这是因为 aspx 页面是从.cs 代码页中的类型继承过来的，而<%#%>标记的作用正是通过在前台显示代码中嵌入访问后台数据的表达式来完成数据绑定。这实际上是通过绑定表达式建立了后台代码与前台页面元素之间的联系，在输出页面流时，系统根据绑定表达式引用后台代码产生的数据计算表达式的值并插入到显示页面的合适位置。因而在绑定表达式中不仅可以引用后台代码中的公有数据成员而且可以引用其公有方法，有兴趣的同学可以自己试验一下。另外，通过表达式绑定数据实际上包含两个步骤：除了为 HTML 元素或服务器空间指定绑定表达式外，还需要在后台代码中显式调用控件的 DataBind()方法，否则绑定过程不能完成，将不能产生预期的显示效果，支持数据绑定的控件都提供了 DataBind()。所以在【例 8-1】后台代码的 Page_Load 方法中才必须包含 Page.DataBind()方法的调用，这里 Page.DataBind()方法会调用页面所有控件及其子控件的 DataBind()方法(包括 HTML 元素)。

对于服务器控件 TextBox1，直接在后台代码中设置了其 TextBox1.Text 属性值来完成数据绑定。这也是一种通用的方法，不仅对单值控件，对于更复杂的数据绑定控件也是适用的。只不过当涉及多值绑定时，需要挨个设置各个显示项的相应属性，当项数较多时会比较繁琐，所以对于多值绑定更常用的方法是通过设置其数据源来完成数据绑定。

8.2.2　列表控件的数据绑定

ASP.NET 中的列表控件有以下几种：DropDownList、ListBox、CheckBoxList、RadioButtonList、BulletedList 等等。虽然它们呈现数据的样式和某些功能有所不同，但本质上都是以数据项列表的形式呈现和组织数据的集合，因此数据绑定的方式也很类似，可以通

过编程的方式为控件对象增加多个数据项，也可以直接在 VWD 2008 环境提供的图形界面中编辑要显示的数据项列表。但由于通常列表控件绑定的是一个数据集合，上述两种方式就比较繁琐了，所以对于列表控件的数据绑定来说，更常用的方式是指定数据源然后调用控件的 DataBind()方法。

下面以 3 种具有代表性的列表控件 DropDownList、CheckBoxList、BulletedList 为例，演示如何通过设置数据源绑定数据。C#中提供的很多集合类都可以作为列表控件的数据源对象，一般来说，实现 IEnumerable、IListSource、IDataSource 和 IHierarchicalDataSource 的类都可以作为数据源。【例 8-2】通过 3 种不同的方式构成 3 种集合类对象作为 3 种列表控件的数据源。

【例 8-2】演示如何通过设置数据源绑定列表控件。

(1) 在 DataBinding 网站中新建一个名为 ListValueBinding 的 web 页并在页面上添加 3 种列表控件，如图 8-2 所示。

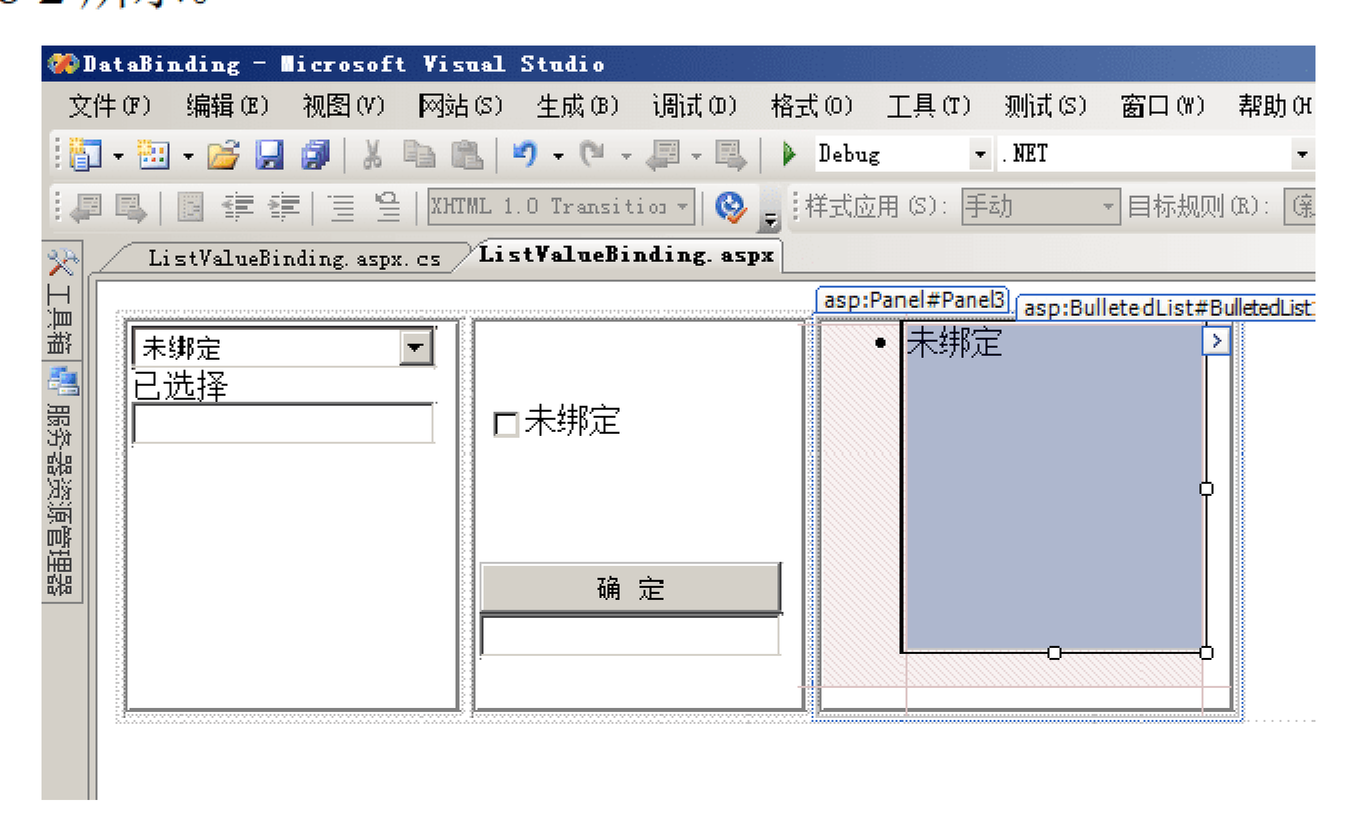


图 8-2　ListValueBinding 设计页面

在页面中添加了如下控件:

- 一个 DropDownList，一个文本框用于显示选择项的值，设置其 AutoPostBack 属性为 True。
- 一个 CheckBoxList，一个确定按钮和一个文本框，控件每一个绑定项显示文本为学生姓名，而值为性别，当按下确定按钮后，将在文本框中显示 CheckBoxList 中所选姓名所对应的性别(键值对)。
- 一个用于显示链接列表的 BulletedList，每一个绑定项都描述一个键值对，代表指向某个网站的链接。显示文本为超链接形式的网站名称，而值为网站 url，因此需要设置控件的 DisplayMode="HyperLink"，并设置 Target="_blank"，代表单击后目标 url 将在新窗口中打开。

页面代码如下：

```
<div>
     <table style="width: 480px;">
  <tr>
 <td>
```

```
            <asp:Panel ID="Panel1" runat="server" Height="190px" Width="160px"
                BorderStyle="Groove">
                <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
                         Height="53px" Width="150px">
              </asp:DropDownList>
              <br />
              <asp:Label ID="Label1" runat="server" Text="已选择" Width="92%"></asp:Label>
              <br />
              <asp:TextBox ID="TextBox1" runat="server" Width="150px"></asp:TextBox>
              <br />
          </asp:Panel>
      </td>
      <td>
          <asp:Panel ID="Panel2" runat="server" Height="190px" Width="160px"
              BorderStyle="Groove" >
              <asp:CheckBoxList ID="CheckBoxList1" runat="server" Height="98px" Width="100%">
                  <asp:ListItem>未绑定</asp:ListItem>
              </asp:CheckBoxList>
              <br />
              <asp:Button ID="Button1" runat="server" Text="确    定" Width="150px" />
              <br />
              <asp:TextBox ID="TextBox2" runat="server" Width="150px" ></asp:TextBox>
              <br />
          </asp:Panel>
      </td>
      <td>
         <asp:Panel ID="Panel3" runat="server" Height="190px" Width="200px"
              BorderStyle="Groove">
              <asp:BulletedList ID="BulletedList1" runat="server"
                  Height="160px" Target="_blank" Width="73%" BulletStyle="Disc">
              </asp:BulletedList>
         </asp:Panel>
      </td>
         </tr>
       </table>
   </div>
```

(2) 在后台页面类中添加如下代码。

```
//定义三种数据源
  //定义并初始化字符串数组
  String[] DataSourceForDDL = new String[] { "张小兵", "李明", "陈飞" };
  //定义哈希表
  Hashtable DataSourceForCBL = new Hashtable(3) ;
  //定义ArrayList
```

```
ArrayList DataSourceForBL = new ArrayList();

protected void Page_Load(object sender, EventArgs e)
{
 if (!IsPostBack)
    {
    //初始化哈希表DataSourceForCBL
    this.DataSourceForCBL.Add("张小兵", "男");
    this.DataSourceForCBL.Add("李明", "女");
    this.DataSourceForCBL.Add("陈飞", "男");
    //初始化DataSourceForBL
    this.DataSourceForBL.Add(new KeyValueClass("网易","http://www.163.com"));
    this.DataSourceForBL.Add(new KeyValueClass("新浪","http://www.sina.com"));
    //为DropDownList绑定数据
    this.DropDownList1.DataSource = this.DataSourceForDDL;
    this.DropDownList1.DataBind();
    //完成绑定后在DropDownList中第一个位置插入一个数据项
    this.DropDownList1.Items.Insert(0, "请选择");
    //为CheckBoxList绑定数据
    this.CheckBoxList1.DataSource = this.DataSourceForCBL;
    //由于哈希表中存储一个键值对的集合并希望在CheckBoxList
    //中处理键值对，因此设定数据源后还需设定DataTextField和DataValueField属性
    this.CheckBoxList1.DataTextField = "key";
    this.CheckBoxList1.DataValueField = "value";
    this.CheckBoxList1.DataBind();
    //为BulletedList绑定数据
    this.BulletedList1.DataSource = this.DataSourceForBL;
    this.BulletedList1.DataTextField = "Name";
    this.BulletedList1.DataValueField = "Url";
    this.BulletedList1.DataBind();
 }
}
```

代码中定义了 3 种集合类对象并进行了初始化：字符串数组 DataSourceForDDL 用作 DropDownList 的数据源，哈希表 DataSourceForCBL 用作 CheckBoxList，ArrayList 对象 DataSourceForBL 用作 BulletedList 的数据源。3 种列表控件的数据项对象都具有以下 2 种属性：DataTextField 和 DataValueField，其中只有 DataTextField 的值会被呈现，因此它们既可以处理单值集合也可以处理键值对数据的集合。

注意：

当处理键值对时，在指定数据源后，要设置上述 2 个属性以指定数据源中对应于键和值的数据域。

(3) 然后再添加一个帮助类 KeyValueClass，其中可以存储一个键值对，用于初始化

DSForBL 对象。

```
/// <summary>
/// 帮助类KeyValueClass的定义，其对象存储一个WebSiteName：WebSiteUrl
/// 的键值对，被添加至作为数据源的ArrayList中
/// </summary>
public class KeyValueClass
{
    private String WebSiteName;
    private String WebSiteUrl;
    public String Name
    {
        get { return WebSiteName;}
        set { WebSiteName = value; }
    }
    public String Url
    {
        get { return WebSiteUrl; }
        set { WebSiteUrl = value; }
    }
    public KeyValueClass(String name,String url)
    {
        this.WebSiteName = name;
        this.WebSiteUrl = url;
    }
}
```

(4) 运行后结果如图 8-3 所示。



图 8-3　数据绑定后的列表控件

当数据绑定完成后就可以通过控件提供的各种事件定制其行为，对数据进行进一步处理。

(5) 为 DropDownList 和确定按钮添加事件处理代码。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
```

```
{
    //清除上次显示内容
    this.TextBox1.Text = "";
    //在文本框中显示所选列表项
    this.TextBox1.Text = this.DropDownList1.SelectedValue;
}
protected void Button1_Click(object sender, EventArgs e)
{
    this.TextBox2.Text = "";
    //循环遍历CheckBoxList中每个项，如果已选择在文本框中显示选中项显示文本和值
    foreach (ListItem li in   CheckBoxList1.Items)
    {
        if(li.Selected)   this.TextBox2.Text += li.Text + ":" + li.Value+ ",";
    }
}
```

(6) 运行效果如图 8-4 所示。



图 8-4　事件处理运行效果

8.3　GridView 控件

GridView 是一个功能强大的数据绑定控件，主要用于以表格形式呈现、编辑关系数据集。对应于关系数据集的结构，GridView 控件以列为单位组织其所呈现的数据，除了普通的文本列，还提供多种不同的内置列样式，例如按钮列、图像列、复选框形式的数据列等等，可以通过设置 GridView 控件的绑定列属性以不同的样式呈现数据，或通过模板列自定义列的显示样式。

在数据绑定时通常将访问关系数据库得到的结果集作为 GridView 控件的数据源，GridView 控件对其所呈现的数据集提供内置的编辑、修改、更新、删除以及分页和排序功能，但是要使用控件的内置数据处理功能，需要使用 ASP.NET 提供的数据源控件(如

SqlDataSource 和 ObjectDataSource)，否则就需要手动编写事件处理程序来实现相应功能。虽然采用数据源控件来连接数据库并处理数据更加方便，但手动编写代码却更加灵活，并且在编写代码的过程中可以更深入地了解 GridView 控件的运行方式，因而也更具有参考意义，因此本章的例子中将采用查询数据库得到的 DataTable 对象作为控件数据源，然后通过编写事件程序的方式来实现数据处理功能，而在下一节中采用数据源控件绑定 FormView 控件并使用内置的数据处理功能。

本节将从以下三个方面介绍 GridView 控件的用法：

- 基本的数据绑定方式
- 各种绑定列和模板列的灵活运用
- 数据的分页和排序

8.3.1　GridView 的数据绑定

【例 8-3】将演示如何进行 GridView 的数据绑定，其基本的数据绑定方式与列表控件类似，首先设置数据源，后调用 DataBind()方法。【例 8-3】在一个 GridView 控件中呈现关系数据集，因此需要访问数据库获取作为控件数据源的结果集，数据库沿用第 7 章的 student 数据库，数据库的访问方式在第 7 章已经详细介绍绍过，这里就不再赘述。

【例 8-3】演示如何如何为 GridView 控件绑定数据源。

(1) 在 DataBinding 网站中新建一个名为 GridViewBingding_1.aspx 的页面，并在页面上添加一个 GridView 控件。如图 8-5 所示。

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Label

图 8-5　GridViewBingding_1.aspx 设计页面

对应页面代码如下：

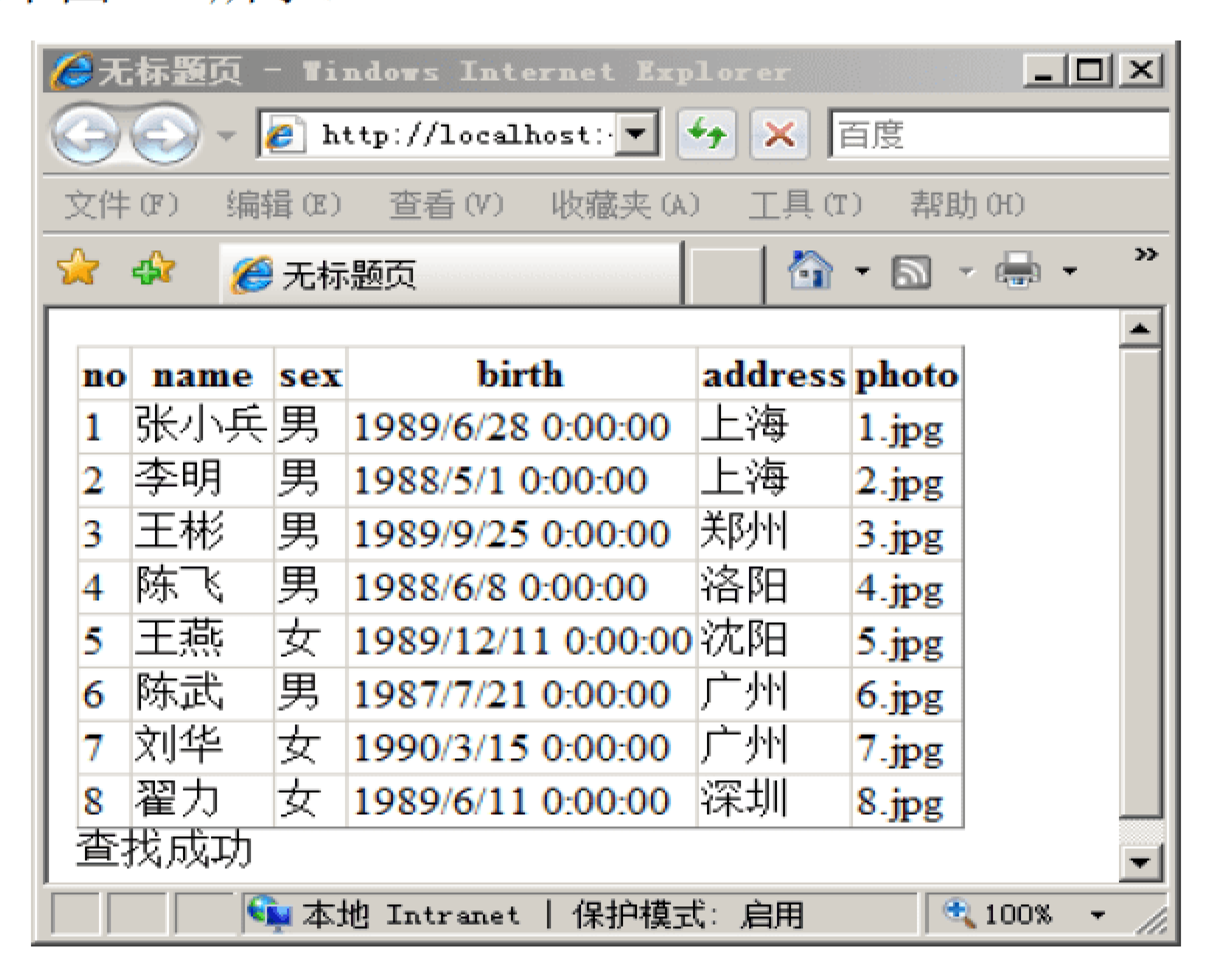
```
<div>
    <asp:GridView ID="GridView1" runat="server"> </asp:GridView>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label><br />
</div>
```

(2) 为页面后台类添加数据绑定代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    //查询student数据库获取结果集ds
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    DataSet ds = new DataSet();
```

```
            using (SqlConnection sqlconn = new SqlConnection(sqlconnstr))
            {
                SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
                sqld.Fill(ds, "tabstudent");
            }
            //以数据集中名为tabstudent的DataTable作为数据源，为控件绑定数据
            GridView1.DataSource = ds.Tables["tabstudent"].DefaultView;
            GridView1.DataBind();
            //label中显示运行状态
            Label1.Text = "查找成功";
    }
```

(3) 运行效果如下图 8-6 所示。



no	name	sex	birth	address	photo
1	张小兵	男	1989/6/28 0:00:00	上海	1.jpg
2	李明	男	1988/5/1 0:00:00	上海	2.jpg
3	王彬	男	1989/9/25 0:00:00	郑州	3.jpg
4	陈飞	男	1988/6/8 0:00:00	洛阳	4.jpg
5	王燕	女	1989/12/11 0:00:00	沈阳	5.jpg
6	陈武	男	1987/7/21 0:00:00	广州	6.jpg
7	刘华	女	1990/3/15 0:00:00	广州	7.jpg
8	翟力	女	1989/6/11 0:00:00	深圳	8.jpg

查找成功

图 8-6　GridViewBingding_1.aspx 数据绑定效果

8.3.2　设定 GridView 的绑定列和模板列

由【例 8-3】可以看到，GridView 的数据绑定方式非常简单，只用几句简单的代码就可以将数据集以表格的形式呈现出来，但这种方式的呈现效果很简陋。事实上可以通过设置 GridView 控件的绑定列属性使其呈现不同的列样式，实现数据的编辑和修改，或编辑模板列定制所需的列样式和功能。

下面在【例 8-4】中将演示如何为 GridView 设置绑定列、调整数据呈现效果、实现数据的编辑和修改功能以及如何通过定义列模板使其呈现自定义样式。在开始实例之前先看看 GridView 提供了哪几种内置的绑定列样式，如何在 VWD 2008 中设置他们以及如何自定义模板列样式。

在 VWD 2008 中可以通过边界任务面板进行列的配置，如图 8-7 所示，单击 GridView 右上角的小箭头打开面板。

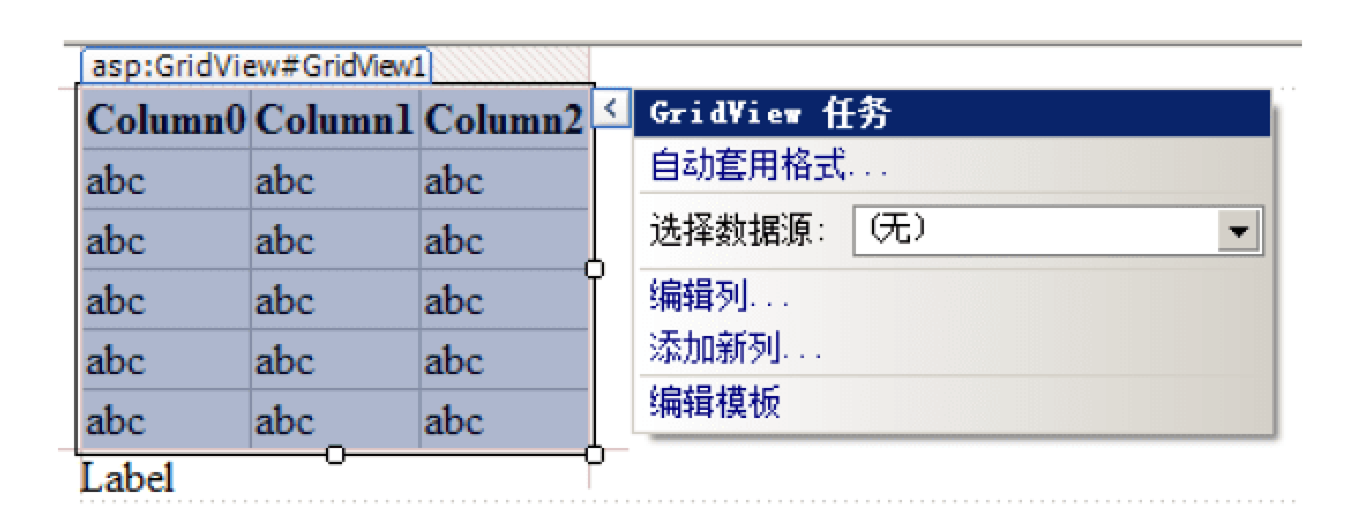


图 8-7　GridView 便捷任务面板

可以看到，面板上包含【自动套用格式】选项，通过【自动套用格式】可以为 GridView 应用一些内置的表格呈现样式。这里着重介绍【编辑列】选项和【编辑模板】选项，其中【编辑列】选项用于设置表格的绑定列数据性，而【编辑模板】选项用于编辑模板列中的显示项的样式。单击图 8-7 的编辑列选项打开设置 GridView 列样式的字段窗口，如图 8-8 所示。

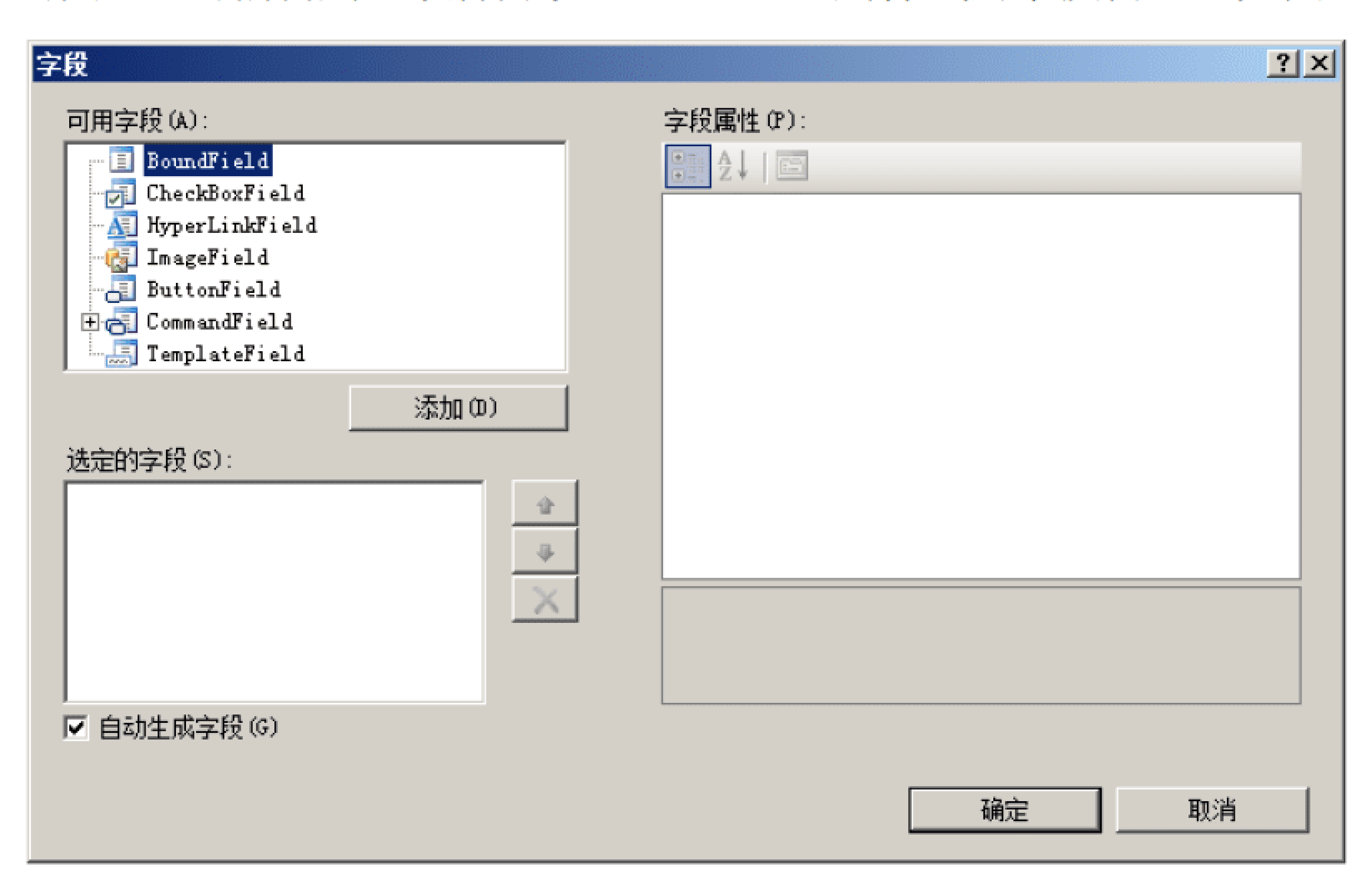


图 8-8　用于编辑 GridView 各个数据列样式的【字段】窗口

图 8-8 中左上角【可用字段】列表列出了可用的绑定列类型，单击【添加】按钮就可以设置 GridView 控件中显示的列及其类型。共有 7 种类型，如下所示。

- BoundField：以文字形式呈现数据的普通绑定列类型。
- CheckBoxField：以复选框形式呈现数据，绑定到该类型的列数据应该具有布尔值。
- HyperLinkField：以链接形式呈现数据，绑定到该类型的列数据应该是指向某个网站或网上资源的地址。
- ImageField：以图片形式呈现数据。
- ButtonField：按钮列，以按钮的形式呈现数据或进行数据的操作。例如删除记录的按钮列。
- CommandField：系统内置的一些操作按钮列，可以实现对记录的编辑、修改、删除等操作。
- TemplateField：模板列中对各个数据项并不给出预设的显示样式，而是通过自定义的方式在项模板中设置数据项的显示方式。

在实际应用的时候，可以根据需要显示的数据类型，选择要绑定的列类型并设置其映射到数据集的字段名称和呈现样式(设置绑定列后 GridView 中将只显示映射列数据，否则系统将默认以 BoundField 类型显示数据源表中的所有列)。如【例 8-3】中作为数据源的 student 表数据，如果想在 GridView 中以 BoundField 类型显示其中字段名为“姓名”的列，可以作如下设置。在图 8-8 所示字段编辑窗口中添加一个 BoundField 列，如图 8-9 所示。

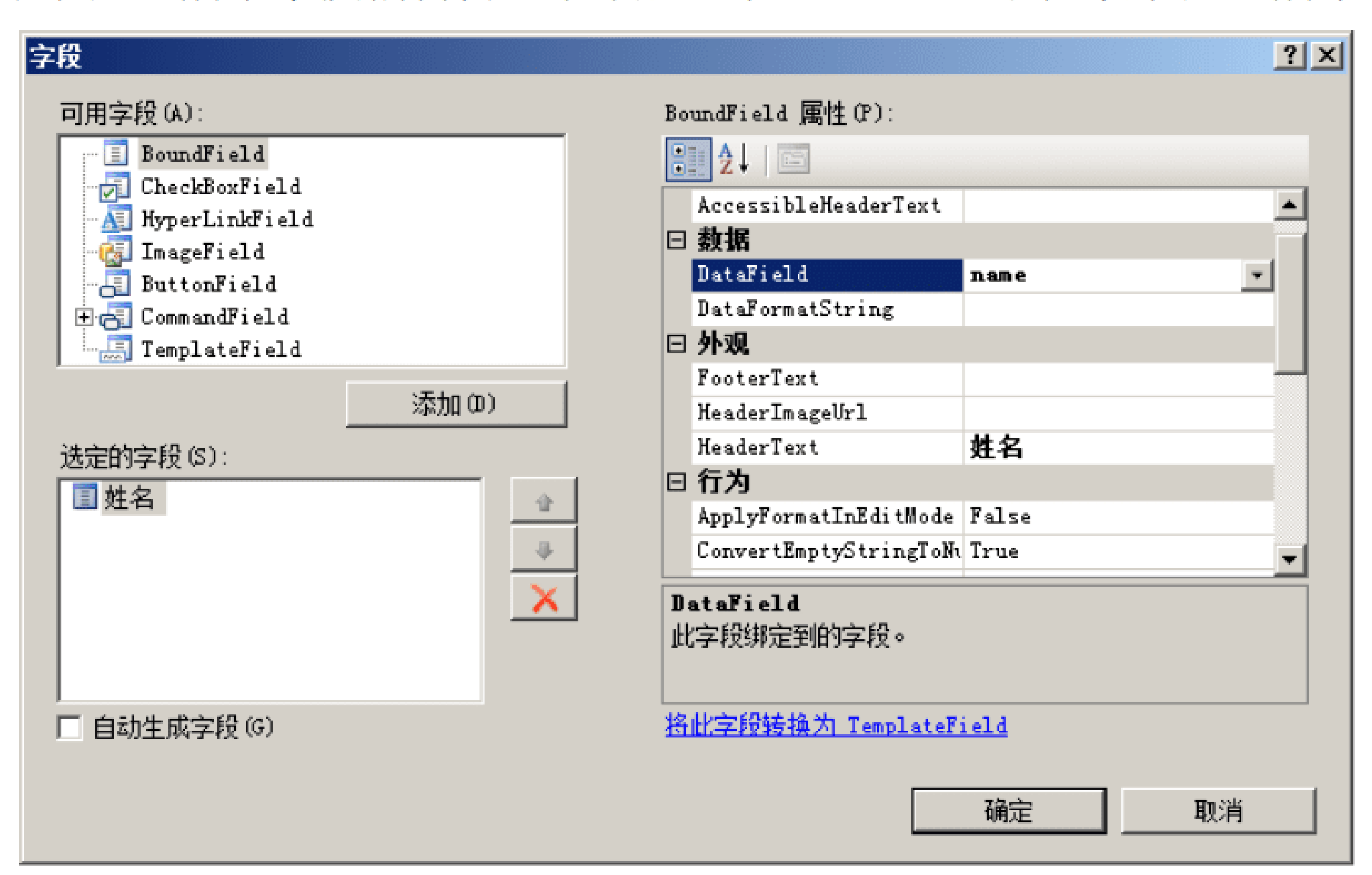


图 8-9　为 GridView 添加绑定列

在右方字段属性编辑框中设置 DataField 数据性为 name，其中 name 对应于作为数据源的 student 表中的 name 字段，通过该属性完成显示列与源之间的数据映射，而 HeaderText 属性表示该字段呈现在 GridView 中时的表头名称，这里设置为“姓名”。在属性编辑框中还可以设置列的显示外观或行为等其他属性，这里就不再赘述。

通过类似的方式就可以为 GridView 控件添加其他类型的绑定列，这里 CommandField 的使用方式稍有特殊。通过 CommandField 类型，并配合事件处理程序就可以在 GridView 中完成数据的编辑、修改、插入等等操作。添加并设置 CommandField 类型的方式如下：展开 CommandField，如图 8-10 所示。

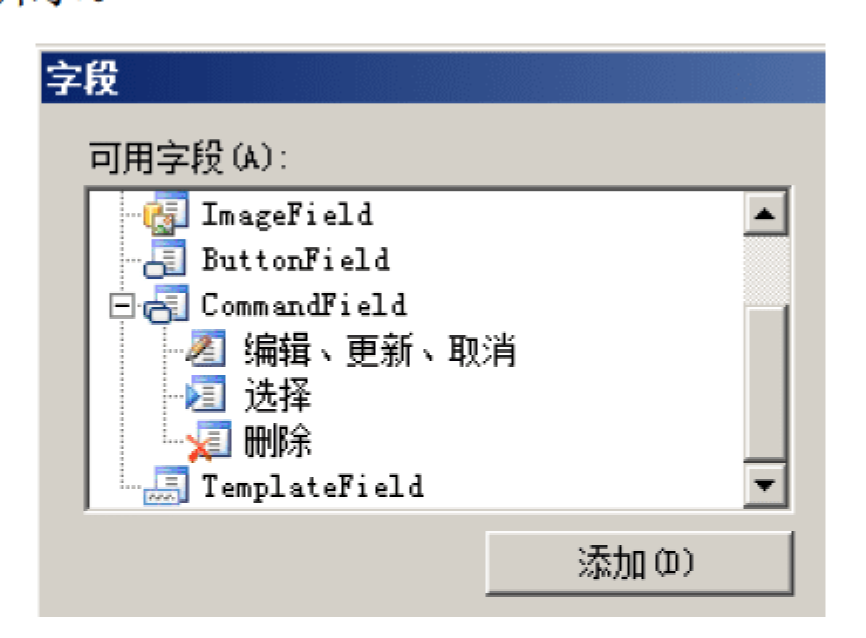


图 8-10　CommandField 类型

可见 CommandField 有 3 种类型可以选择，不同的类型意味着在 CommandField 列显示不同的命令按钮，例如选择【编辑、更新、取消】，列样式如图 8-11 所示。

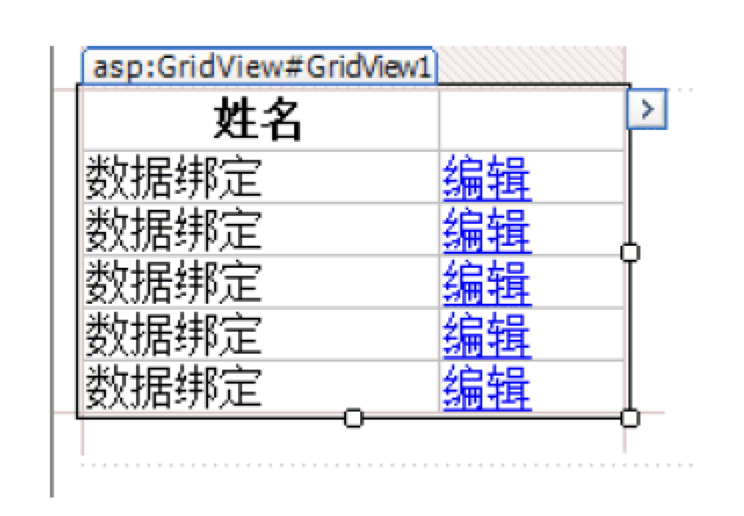


图 8-11　添加【编辑、更新、取消】列

运行时单击【编辑】按钮，列中的编辑按钮将会被替换为两个按钮【更新、取消】，因此列的运行时实际上包含了 3 个命令按钮，单击按钮所发生的行为需要通过设置相应的事件程序完成，由于 CommandField 类型是一种控件内置的用于编辑数据的绑定类型，因此其事件在 GridView 控件的属性窗口中设置，GridView 空间的属性窗口并选择事件列表如图 8-12 所示。

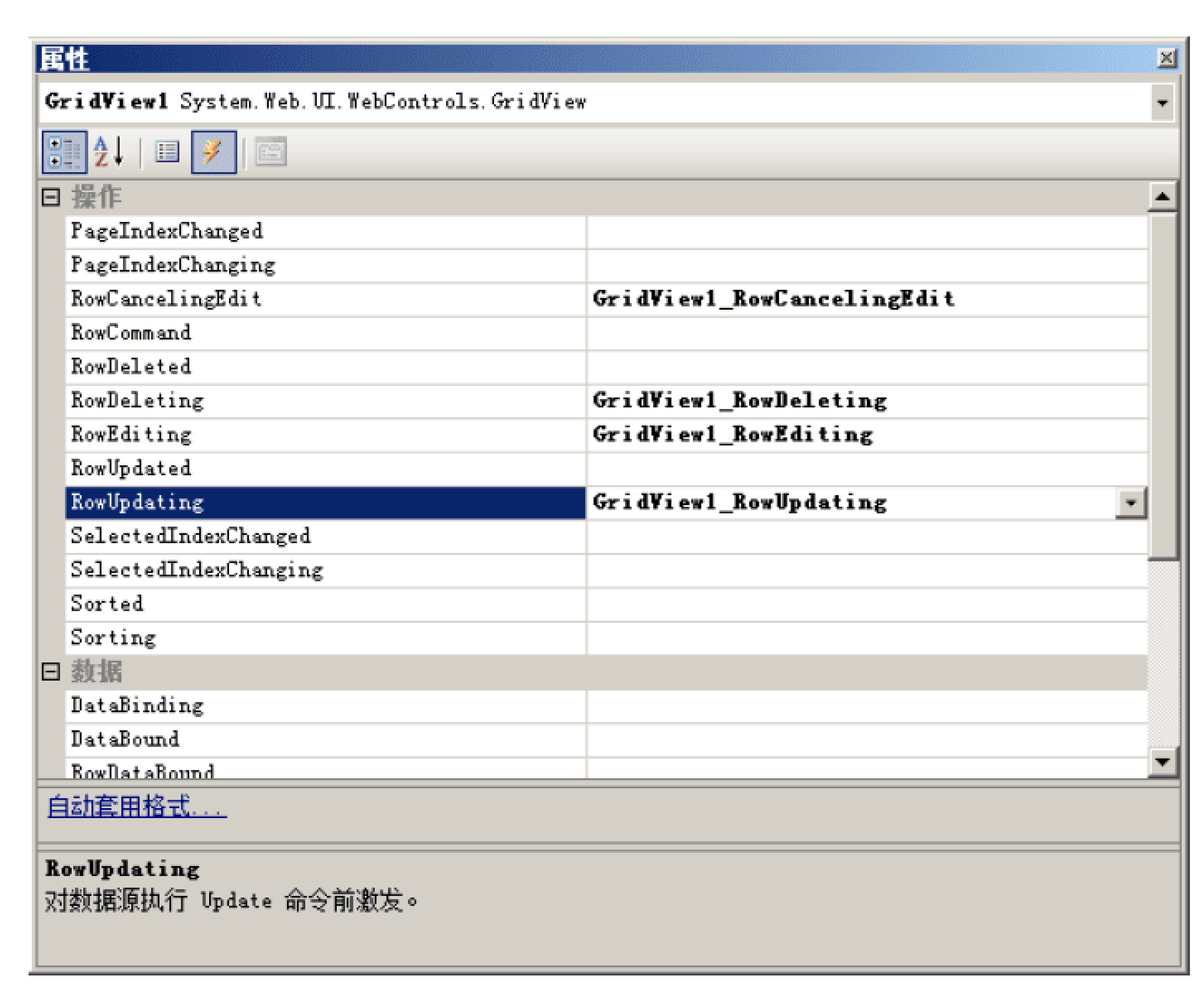


图 8-12　GridView 的事件编辑窗口

其中的 RowEditing、RowUpdating、RowDeleting、RowCancelingEdit 事件分别在编辑、更新、删除、取消按钮被单击时触发。通过为这些事件添加相应的处理程序就可以完成数据的编辑和修改功能。

对于 TemplateField 类型，需要先编辑模板来定义列中各个项的显示样式，然后根据自定义模板绑定模板列，系统将根据模板中定义的样式呈现数据。GridView 中自定义模板的方式与 DataList 或 Repeater 控件的模板定义方式类似，放在后续章节介绍。

【例 8-4】演示为 GridView 设置绑定列来显示 student 表数据的完整过程，例【8-4】中将包含 4 个 BoundField 列，1 个 CommandField 列和 1 个 ButtonField 列，这里将为各个命令按钮添加事件处理程序完成数据的编辑、更新和删除功能。

【例 8-4】演示为 GridView 控件设置绑定列。

(1) 在 DataBinding 网站中新建一个名为 GridViewBingding_2.aspx 的页面，在页面上添

加一个 GridView 控件，为其添加如下绑定列并设置数据映射，如图 8-13 所示。

- 学号列：BoundField 类型，绑定字段 no。
- 姓名列：BoundField 类型，绑定字段 name。
- 出生日期列：BoundField 类型，绑定字段 birth。
- 地址列：BoundField 类型，绑定字段 address。
- 编辑列：CommandField 类型，子类型为编辑、更新、取消，实现数据的编辑和更新。
- 删除按钮列：ButtonField 类型，实现记录的删除。

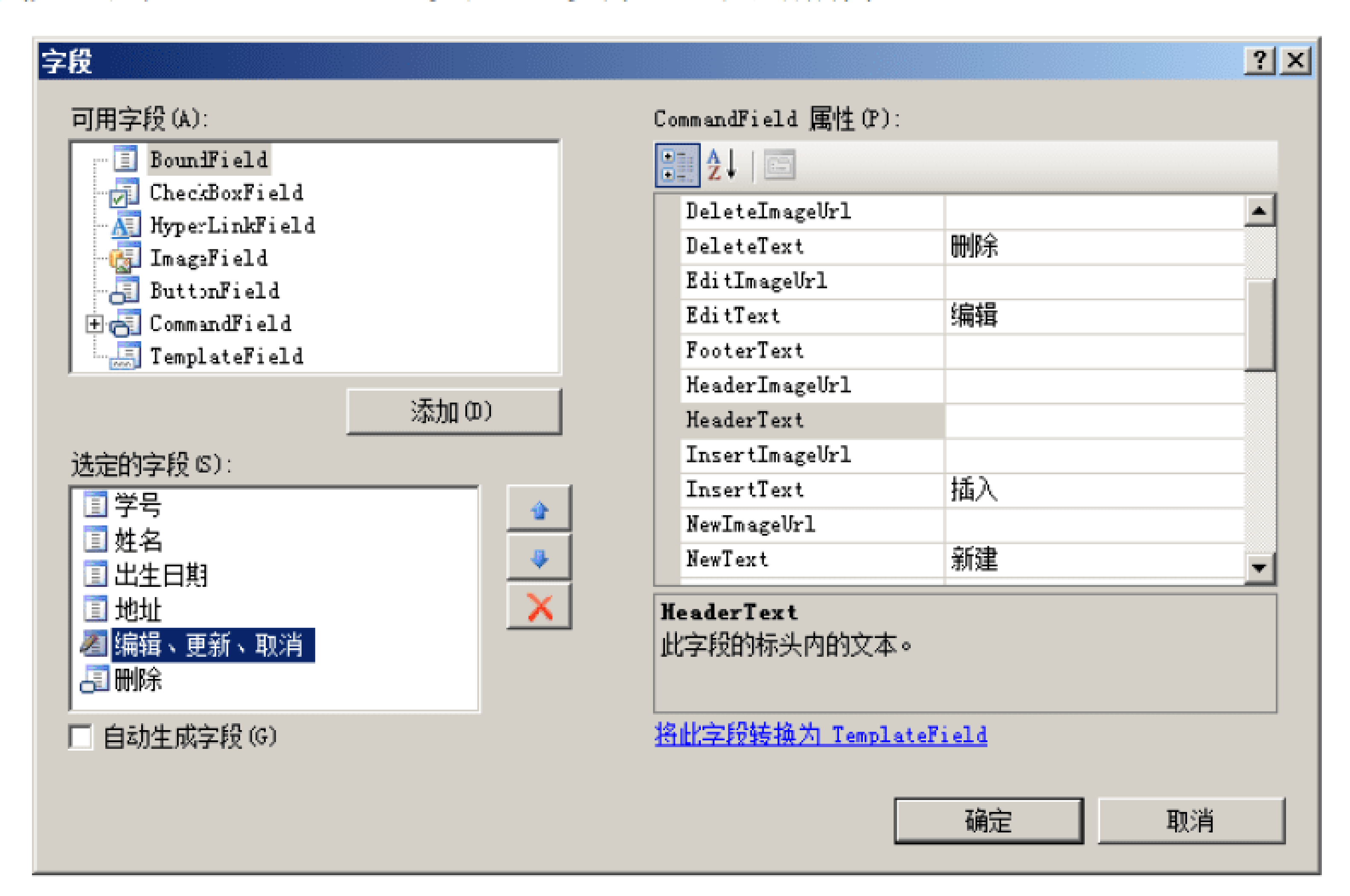


图 8-13　为 GridView 控件添加并设置绑定列

页面设计效果如图 8-14 所示。

asp:GridView#GridView1

学号	姓名	出生日期	地址		
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除

图 8-14　GridViewBingding_2.aspx 设计页面

页面代码如下：

```
<div>
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
  OnRowCancelingEdit="GridView1_RowCancelingEdit"
        OnRowEditing="GridView1_RowEditing" OnRowUpdating="GridView1_RowUpdating"
        DataKeyNames="no" OnRowDeleting="GridView1_RowDeleting" Height="185px"
        Width="478px">
        <Columns>
            <asp:BoundField DataField="no" HeaderText="学号" ReadOnly="True" />
            <asp:BoundField DataField="name" HeaderText="姓名" />
```

```
                <asp:BoundField DataField="birth" HeaderText="出生日期" />
                <asp:BoundField DataField="address" HeaderText="地址" />
                <asp:CommandField InsertVisible="False" ShowEditButton="True" />
                <asp:ButtonField ButtonType="Button" CommandName="delete" Text="删除" />
            </Columns>
        </asp:GridView>
</div>
```

(2) 在页面后台类中添加如下代码进行数据绑定。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack) bindgrid();
}
void bindgrid()
{
   //查询student数据库获取结果集ds
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    DataSet ds = new DataSet();
    using (SqlConnection sqlconn = new SqlConnection(sqlconnstr))
    {
      SqlDataAdapter sqld = new SqlDataAdapter("select no,name,birth,address from student", sqlconn);
      sqld.Fill(ds, "tabstudent");
    }
   //以数据集中名为tabstudent的DataTable作为数据源，为控件绑定数据
   GridView1.DataSource = ds.Tables["tabstudent"].DefaultView;
   GridView1.DataBind();
}
```

(3) 为命令按钮列绑定事件处理方法，如图 8-15 所示。

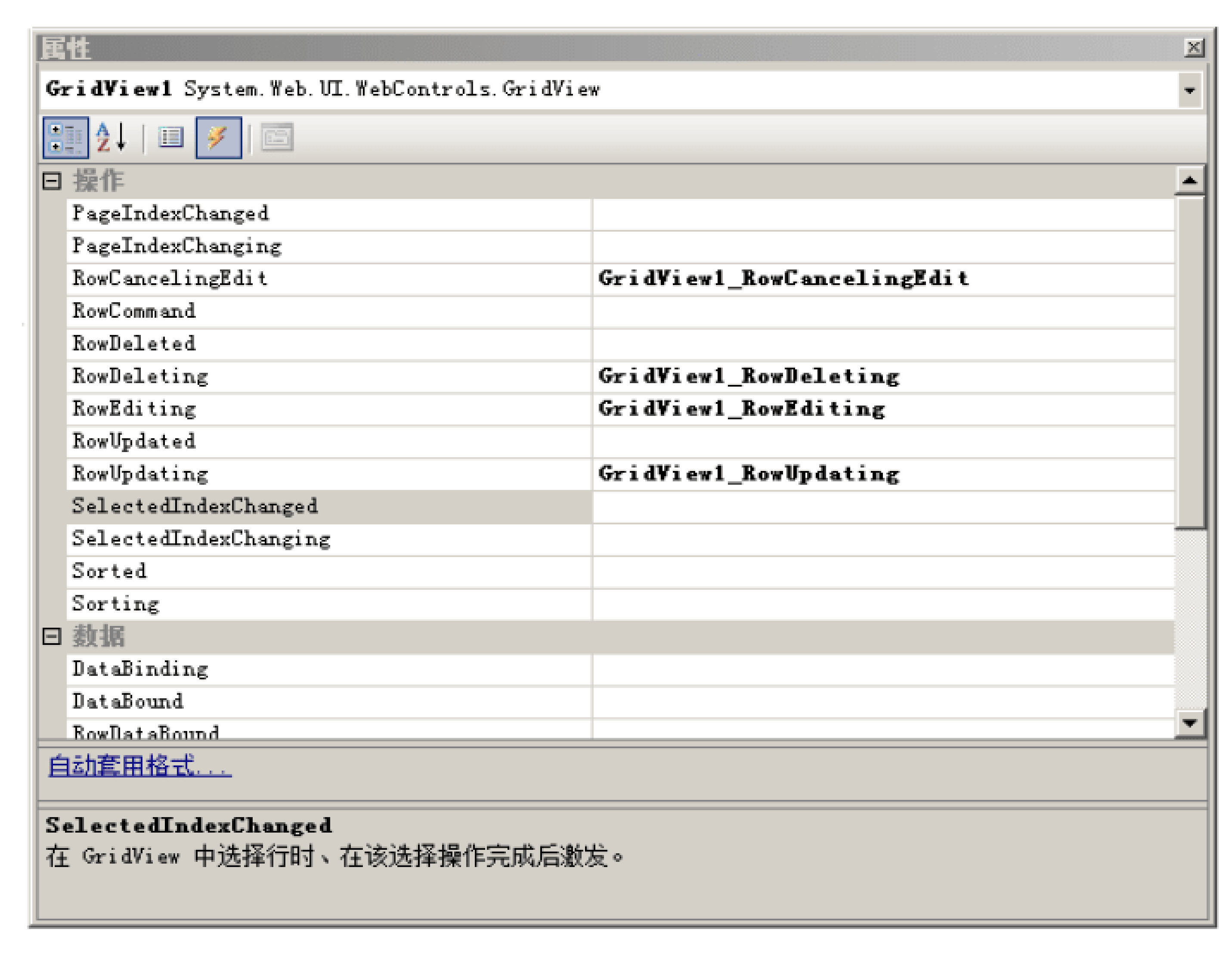


图 8-15　为命令事件设置事件处理方法

(4) 在页面后台事件处理方法中添加事件处理代码。

```
  protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
  {
      GridView1.EditIndex = e.NewEditIndex;
      bindgrid();
  }
  protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
  {
     string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString; ;
     SqlConnection sqlconn = new SqlConnection(sqlconnstr);
        //提交行修改
     sqlconn.Open();
     SqlCommand Comm = new SqlCommand();
     Comm.Connection = sqlconn;
     Comm.CommandText = "update student set name=@name,birth=@birth,address=@address where
no=@no";
     Comm.Parameters.AddWithValue("@no", GridView1.DataKeys[e.RowIndex].Value.ToString());
     Comm.Parameters.AddWithValue("@name",
((TextBox)GridView1.Rows[e.RowIndex].Cells[1].Controls[0]).Text);
     Comm.Parameters.AddWithValue("@birth",
((TextBox)GridView1.Rows[e.RowIndex].Cells[2].Controls[0]).Text);
     Comm.Parameters.AddWithValue("@address",
((TextBox)GridView1.Rows[e.RowIndex].Cells[3].Controls[0]).Text);
     Comm.ExecuteNonQuery();
     sqlconn.Close();
     sqlconn = null;
     Comm = null;
     GridView1.EditIndex = -1;
     bindgrid();
 }
 protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
 {
     GridView1.EditIndex = -1;
     bindgrid();
 }
 protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
 {
     //设置数据库连接
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString; ;
   SqlConnection sqlconn = new SqlConnection(sqlconnstr);
   sqlconn.Open();
   //删除行处理
   String sql = "delete from student where no='" + GridView1.DataKeys[e.RowIndex].Value.ToString() + "'";
   SqlCommand Comm = new SqlCommand(sql, sqlconn);
```

```
Comm.ExecuteNonQuery();
sqlconn.Close();
sqlconn = null;
Comm = null;
GridView1.EditIndex = -1;
bindgrid();       }
```

(5) 页面运行效果如图 8-16 所示。

学号	姓名	出生日期	地址		
1	张小兵	1989/6/26 0:00:00	上海	编辑	删除
2	李明	1988/5/1 0:00:00	上海	编辑	删除
3	王彬	1989/9/25 0:00:00	郑州	编辑	删除
4	陈飞	1988/6/8 0:00:00	洛阳	编辑	删除
5	王燕	1989/12/11 0:00:00	沈阳	编辑	删除
6	陈武	1987/7/21 0:00:00	广州	编辑	删除
7	刘华	1990/3/15 0:00:00	广州	编辑	删除
8	翟力	1989/6/11 0:00:00	深圳	编辑	删除

图 8-16　GridViewBingding_2.aspx 运行效果

(6) 单击编辑按钮后出现编辑和更新界面，如图 8-17 所示。

学号	姓名	出生日期	地址		
1	张小兵	1989/6/26 0:00:00	上海	更新 取消	删除
2	李明	1988/5/1 0:00:00	上海	编辑	删除
3	王彬	1989/9/25 0:00:00	郑州	编辑	删除
4	陈飞	1988/6/8 0:00:00	洛阳	编辑	删除
5	王燕	1989/12/11 0:00:00	沈阳	编辑	删除
6	陈武	1987/7/21 0:00:00	广州	编辑	删除
7	刘华	1990/3/15 0:00:00	广州	编辑	删除
8	翟力	1989/6/11 0:00:00	深圳	编辑	删除

图 8-17　GridViewBingding_2.aspx 的编辑效果

8.3.3　GridView 的排序

GridView 控件提供了用于实现排序功能的接口，通过设置相关属性并实现排序事件的处理代码就可以完成排序功能。这里将以【例 8-4】提供的界面的基础上实现排序功能。

【例 8-5】演示为 GridView 控件实现排序。

(1) 在【例 8-4】中 GridViewBingding_2.aspx 页面中设置 GridView 控件属性

AllowSorting=True。如图 8-18 所示。

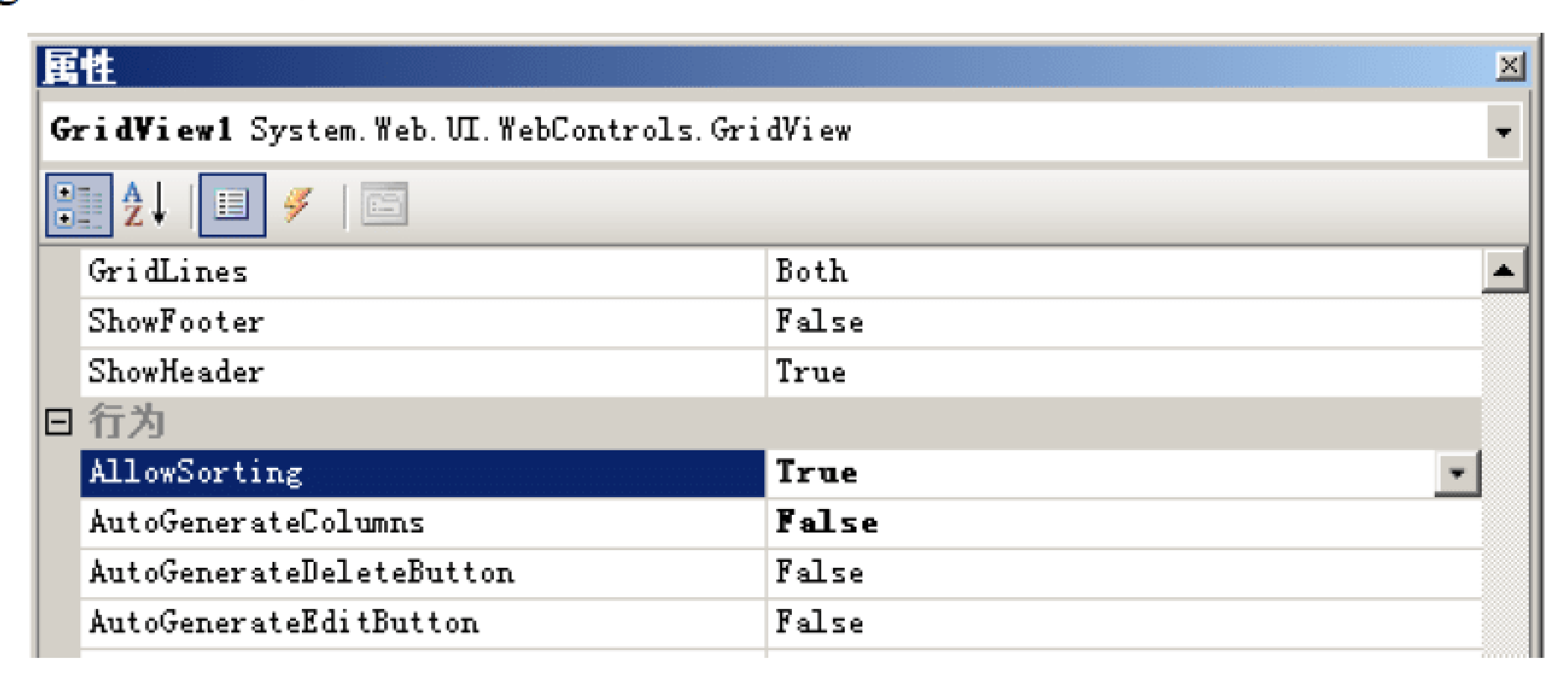


图 8-18　设置 AllowSorting 属性

除了 AllowSorting 属性，还必须设置作为排序关键字的列的 SortExpression 属性，这是因为 GridView 中可以包含按钮列，按钮列一般并不映射到某个数据字段，而排序必须以某个字段作为排序关键字才能完成。

(2) 在 GridView 控件的便捷任务面板中打开编辑列窗口，选择可以作为排序关键字的列，设置其 SortExpression 属性为排序字段名，如图 8-19 所示。

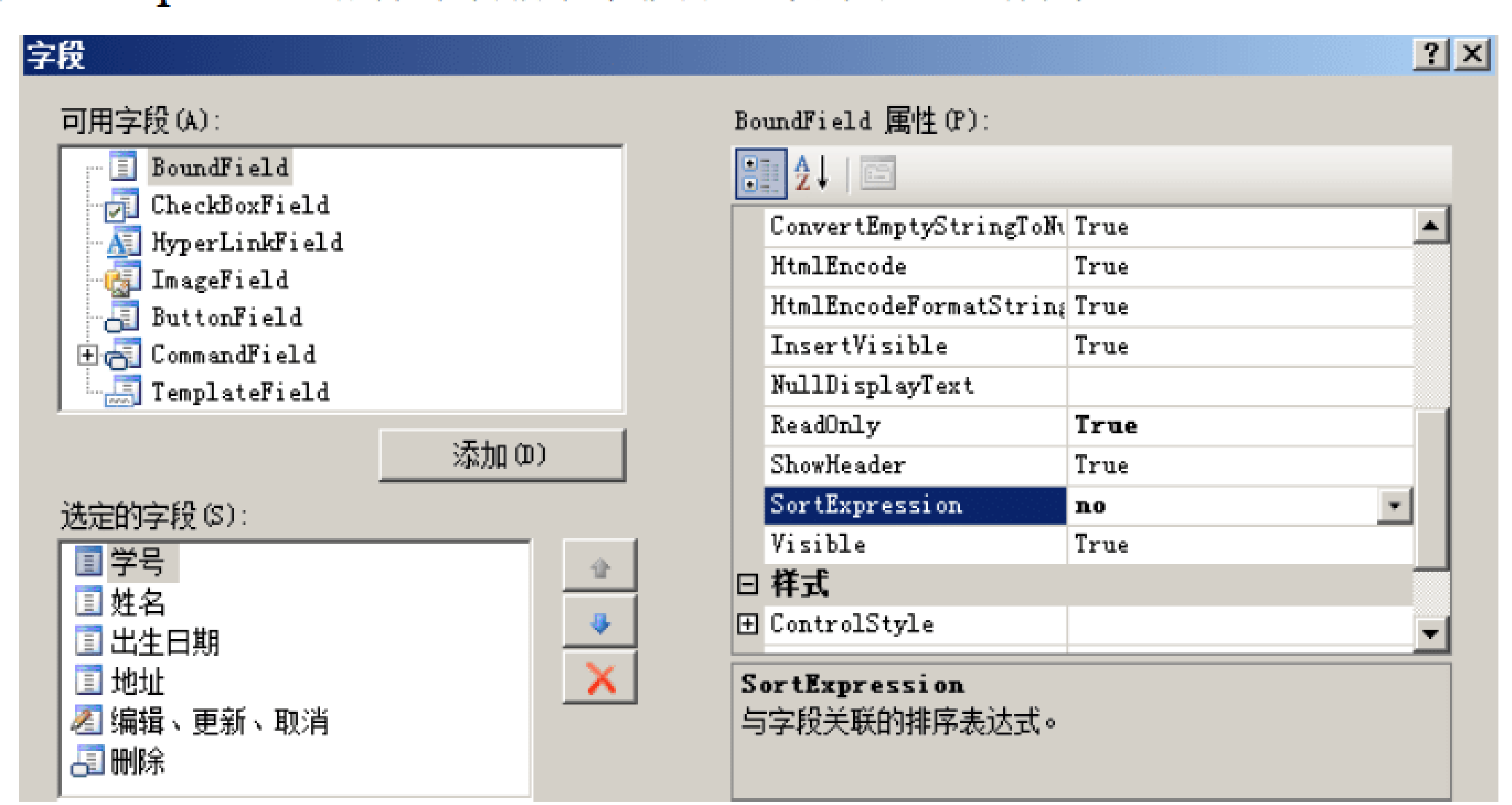


图 8-19　设置 SortExpression 属性

这时作为排序关键字的列的列名变为超链接样式，如图 8-20 所示。

学号	姓名	出生日期	地址		
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	数据绑定	编辑	删除

图 8-20　设置排序属性后的控件样式

(3) 为 GridView 控件设置排序事件处理方法，如图 8-21 所示。

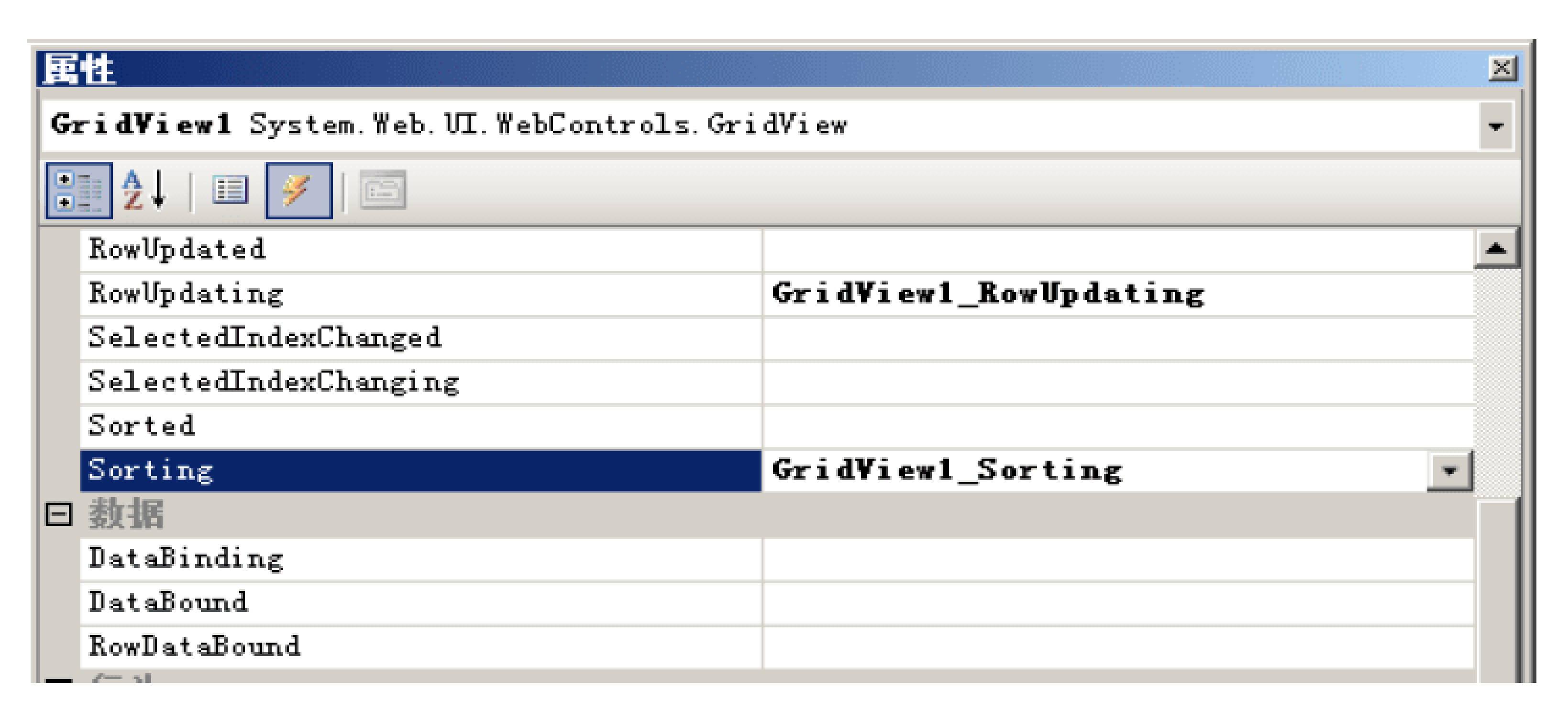


图 8-21　为控件设置排序事件处理方法

GridView 的排序功能通过响应排序事件在后台生成已排序的数据源后重新绑定数据来完成，因此需要事件相应代码中需要获取排序字段名和排序方式(升序、降序)，然后据此对数据源进行排序后重新绑定数据。

(4) 为排序事件处理方法添加代码如下，代码中用一个 ViewState["SortDirection"]来记录当前的排序顺序，用一个 ViewState["SortExpression"]记录作为排序关键字的字段名，然后重新绑定数据。

```
protected void GridView1_Sorting(object sender, GridViewSortEventArgs e)
{
    if(ViewState["SortDirection"] == null) ViewState["SortDirection"] = "DESC";
    if (ViewState["SortDirection"] .ToString() == "ASC")
        ViewState["SortDirection"] = "DESC";
    else
        ViewState["SortDirection"] = "ASC";
    ViewState["SortExpression"] = e.SortExpression;
    this.DataBind();
}
```

修改 DataBind()代码如下，使其根据 ViewState["SortDirection"]的值生成排序后的 DataView 对象作为数据源。

```
void bindgrid()
{
      string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString; ;
      DataSet ds = new DataSet();
      using (SqlConnection sqlconn = new SqlConnection(sqlconnstr))
      {
          SqlDataAdapter sqld = new SqlDataAdapter("select no,name,birth,address from student", sqlconn);
          sqld.Fill(ds, "tabstudent");
      }
      //判断是否已经进行排序，如果是则按照ViewState中存储的信息生成排序后的DataView对象
      if (ViewState["SortDirection"] == null)
          GridView1.DataSource = ds.Tables["tabstudent"].DefaultView;
```

```
            else
            {
                DataView SortedDV = new DataView(ds.Tables["tabstudent"]);
                SortedDV.Sort = ViewState["SortExpression"].ToString() + " " +
ViewState["SortDirection"].ToString();
                GridView1.DataSource = SortedDV;
            }
            GridView1.DataBind();
        }
```

(5) 排序效果如图 8-22 所示。



图 8-22　GridView 排序效果

8.3.4　GridView 的分页

GridView 控件提供了内置的分页功能，绑定数据后只要设置分页相关属性系统即可自动完成分页功能，只需在分页导航按钮的单击事件处理方法中添加代码，设置当前要显示的页索引并重新绑定数据即可。

【例 8-6】演示为 GridView 控件实现分页。

(1) 在 DataBinding 网站中新建一个名为 GridViewBingding_3.aspx 的页面，在页面上添加一个 GridView 控件，并添加用于显示分页信息的 Label，页面设计如图 8-23 所示。

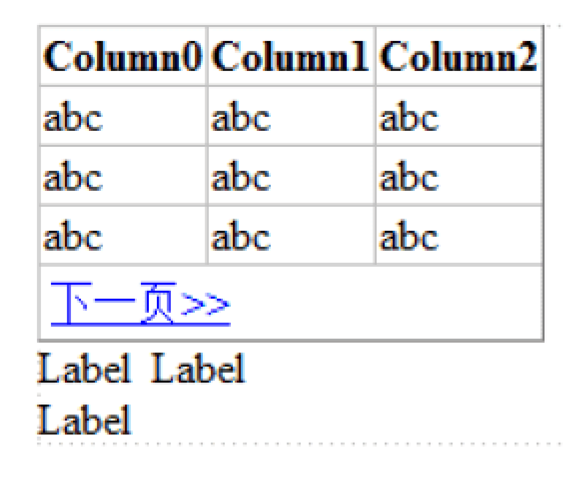


图 8-23　GridViewBingding_3.aspx 设计页面

(2) 页面代码如下。

```
<div>
```

```
            <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
OnPageIndexChanging="GridView1_PageIndexChanging"
                PageSize="3" OnDataBound="GridView1_DataBound">
                <PagerSettings Mode="NextPrevious" NextPageText="下一页&gt;&gt;"
PreviousPageText="&lt;&lt;上一页" />
            </asp:GridView>
            <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
             
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label><br />
            <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label><br />
</div>
```

(3) 在后台添加数据绑定代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack) bindgrid();
}
void bindgrid()
{
   //查询student数据库获取结果集ds
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    DataSet ds = new DataSet();
    using (SqlConnection sqlconn = new SqlConnection(sqlconnstr))
    {
      SqlDataAdapter sqld = new SqlDataAdapter("select no,name,birth,address from student", sqlconn);
      sqld.Fill(ds, "tabstudent");
    }
   //以数据集中名为tabstudent的DataTable作为数据源，为控件绑定数据
   GridView1.DataSource = ds.Tables["tabstudent"].DefaultView;
   GridView1.DataBind();      }
```

(4) 打开 GridView 数据属性设置窗口，为其设置分页相关属性，如图 8-24 所示。

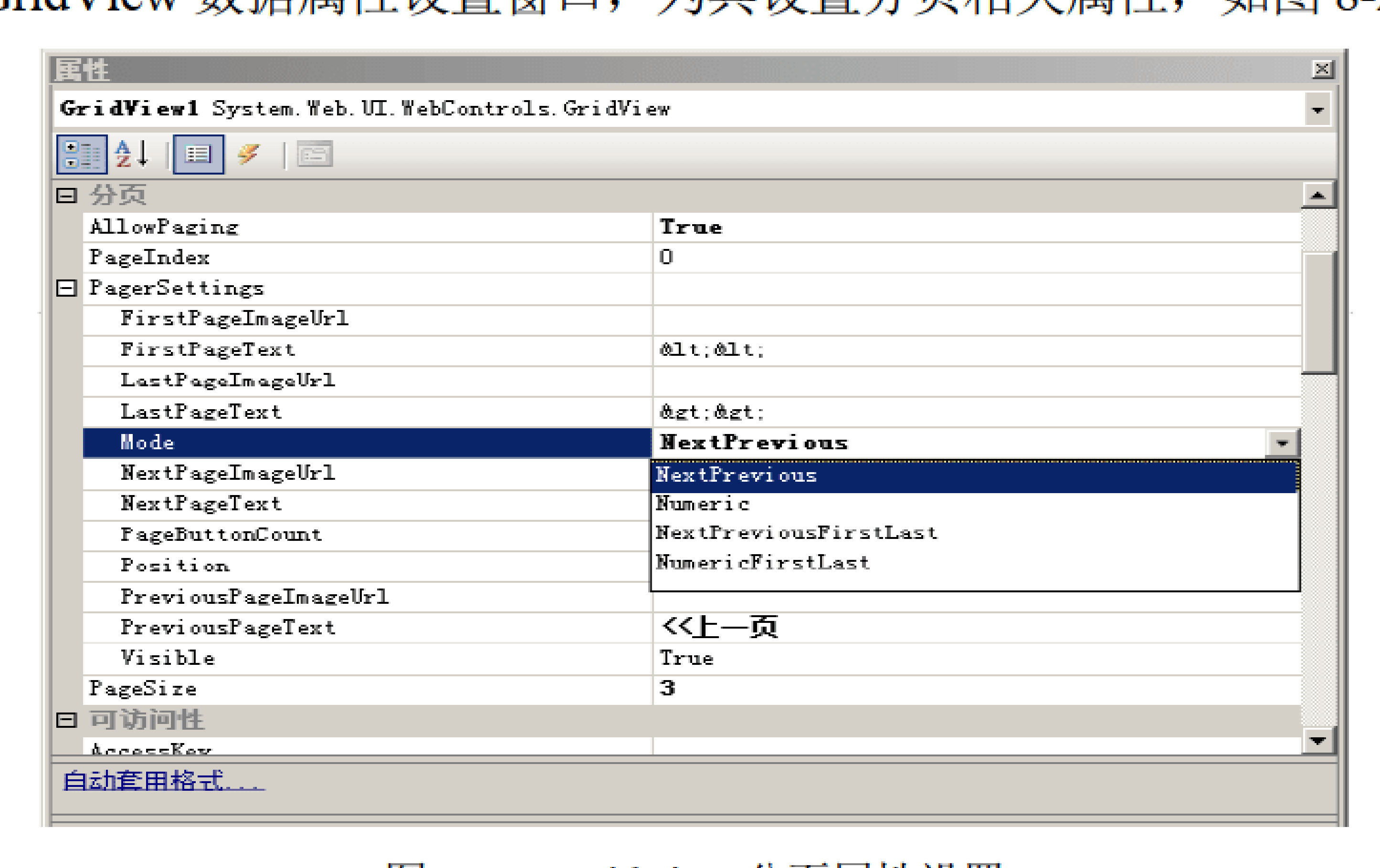


图 8-24　GridView 分页属性设置

分页的设置主要有三个属性。

- AllowPaging：设置是否打开分页功能。
- PageIndex：当前显示的页索引。
- PageSize：设置每页包含的最大项数。

除了上述 3 种分页属性，还可展开 PageSettings 子项，在其中设置分页模式、分页按钮的显示文本等等分页后的控件样式。其中 Mode 属性设置分页模式，共有 4 种可选模式，这里选择 NextPrevieus 模式。

(5) 设置完分页属性后就可以为分页导航按钮设置分页事件处理方法，如图 8-25 所示。

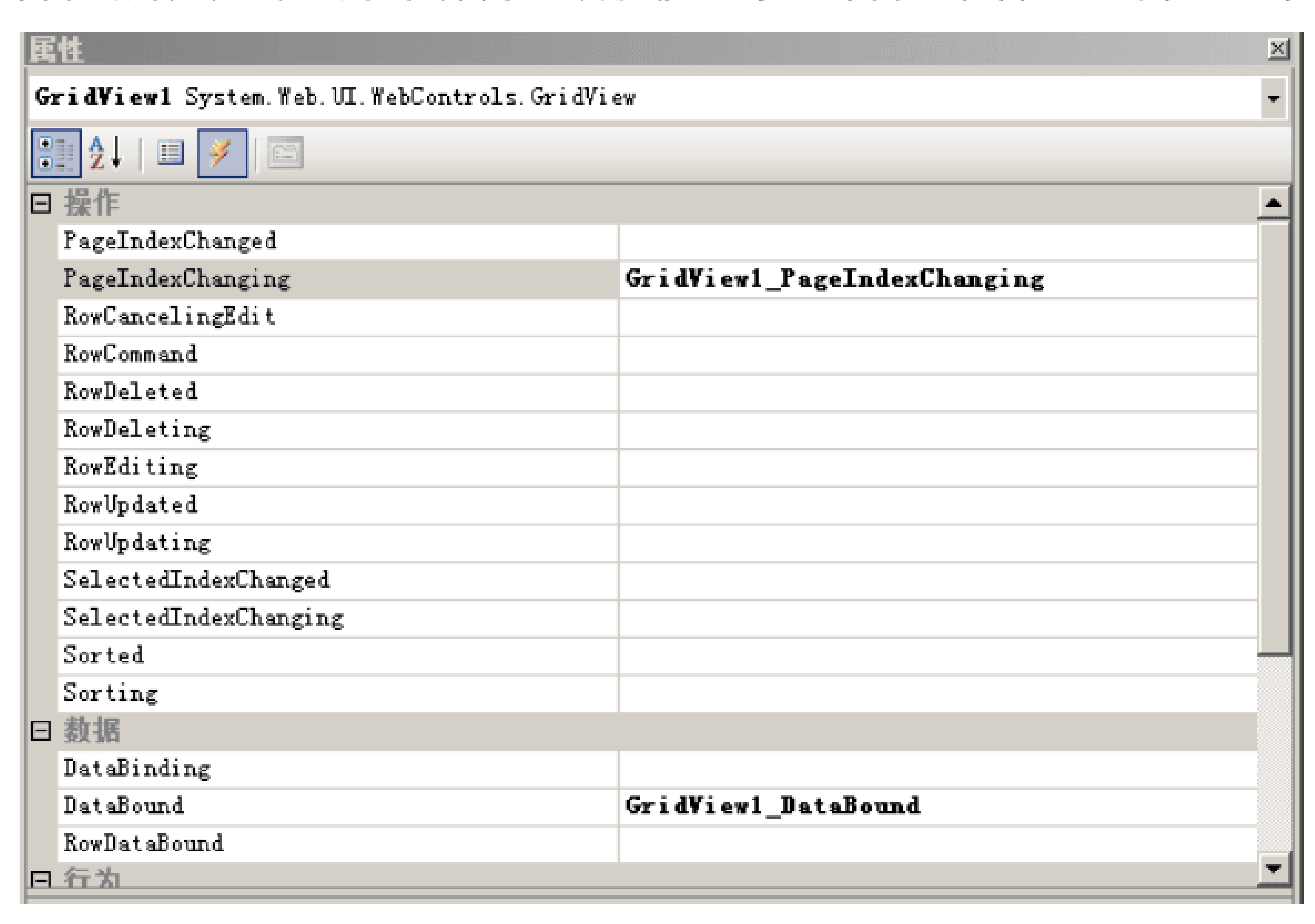


图 8-25　设置分页事件处理方法

图 8-25 中为 PageIndexChanging 事件设置了事件处理方法，该事件在分页导航按钮被单击时触发，并返回导航按钮所指示的，也就是控件中要显示的页的索引，在其事件处理方法中根据该索引设置要显示的页并重新绑定数据即可完成分页。另外还设置了 DataBound 事件的处理方法，用于在分页时重新绑定数据后设置 Label 控件显示分页信息，以及总共的页数、当前页数。

(6) 为事件处理方法添加代码如下。

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    //设置要显示的页的索引并重新绑定数据
    GridView1.PageIndex = e.NewPageIndex;
    bindgrid();
}
protected void GridView1_DataBound(object sender, EventArgs e)
{
    //分页数据绑定前设置当前页信息
    Label2.Text = "共" + (GridView1.PageCount).ToString() + "页";
    Label1.Text = "第" + (GridView1.PageIndex + 1) .ToString() + "页";
    Label3.Text = string.Format("总页数：{0}，当前页：{1}", GridView1.PageCount, GridView1.PageIndex
```

```
+ 1);
    }
```

(7) 运行效果如下图 8-26 所示。

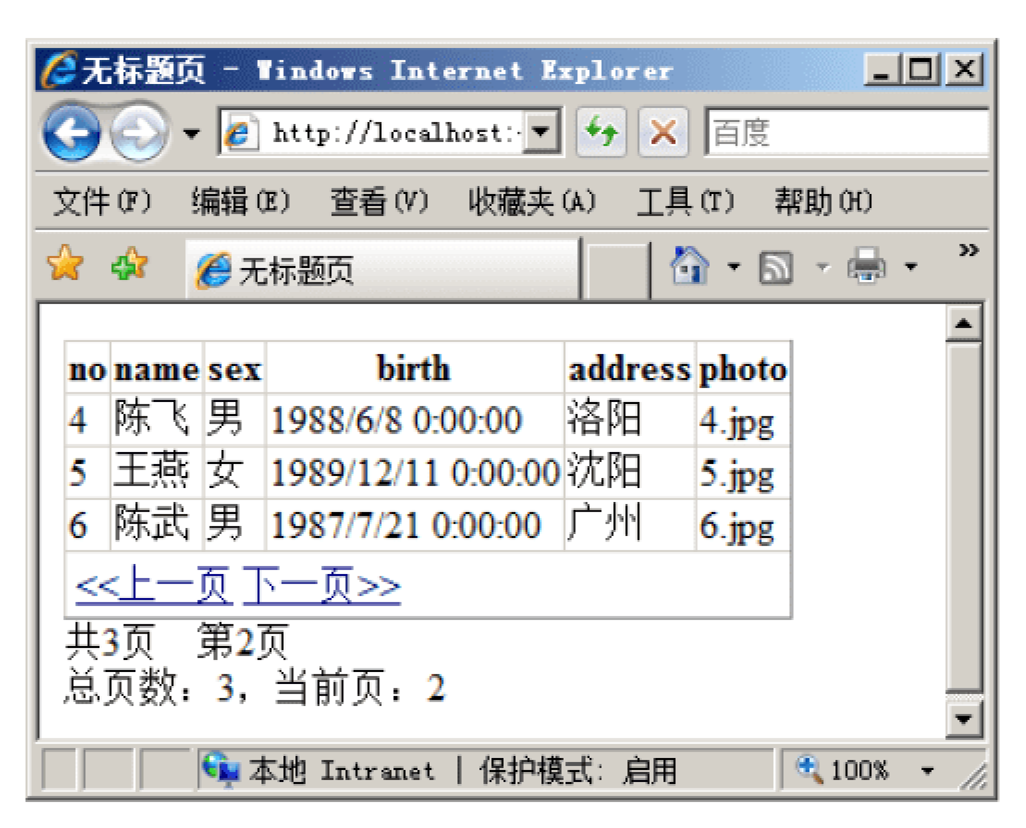


图 8-26　GridViewBingding_3.aspx 运行效果

8.4　DataList 和 FormView 控件

本节将介绍另外 2 种较为复杂的数据绑定控件: DataList 和 FormView 控件，与 GridView 一样，这 2 种服务器控件也用于呈现关系数据库集，但它们不像 GridView 控件以固定的表格样式显示数据，而必须以自定义模板的方式定制数据的呈现样式，这与 Gridview 的自定义模板列非常类似(因此本书在 8.3 节并没有详细介绍 GridView 模板列用法)。DataList 和 FormView 控件以项为单位组织和呈现数据(GridView 以列为单位)，每一项对应于关系数据集的一条记录(行)，通过定义和设置不同的项模板定制每一项的显示样式，绑定数据后控件将按照项模板重复显示数据源的每条记录。呈现数据时 3 种控件对项的显示布局各不相同。DataList 控件提供 2 种页面布局：Table 和 Flow，在 Table 模式下在一个行列表中重复每个数据项，可以通过相关属性控制其按行显示或按列显示并设置行(列)中包含的最大项数；Flow 模式下在一行或者一列中重复显示数据项。FormView 控件默认每页显示一个数据项，通过分页导航访问每条记录。

在 DataList 和 FormView 控件中可以实现对关系数据集的编辑、更新、插入、删除和分页等数据处理功能。DataList 和 FormView 控件针对数据源控件提供内置的数据处理功能，只需某些配置即可自动完成，而针对其他类型的数据源公开特定的属性和事件通过编写代码来实现。

通过 8.3 节对 GridView 控件的详细介绍，可以看出，复杂数据绑定控件的用法不外以下三个方面：

- 数据的绑定与呈现
- 数据的编辑、修改、添加、删除
- 数据的分页和排序

下面通过实例分别加以介绍，由于 3 种控件的用法类似，这里只针对 DataList 实现数据的呈现与绑定，针对 FormView 实现数据的编辑、增、删、改及分页。

另外还有一种基于项模板数据绑定控件：Repeater，它不提供任何内置的显示布局和内置的数据处理功能，只是按照定义好的项模板简单地重复数据。由于 Repeater 控件，通常只是用于以同一样式重复显示数据记录，而其项模板样式的定义和使用方法与 FormView 和 DataList 极为相似，因此不再提供实例，读者可以自行演练。

8.4.1　DataList 的数据绑定

DataList 控件中通过自定义模板设置数据的显示样式，它支持如下模板类型。

- ItemTemplate：包含一些 HTML 元素和控件，将为数据源中的每一行呈现一次这些 HTML 元素和控件。
- AlternatingItemTemplate：包含一些 HTML 元素和控件，将为数据源中的每两行呈现一次这些 HTML 元素和控件。通常，可以使用此模板来为交替行创建不同的外观，例如指定一个与在 ItemTemplate 属性中指定的颜色不同的背景色。
- SelectedItemTemplate：包含一些元素，当用户选择 DataList 控件中的某一项时将呈现这些元素。通常，可以使用此模板来通过不同的背景色或字体颜色直观地区分选定的行。还可以通过显示数据源中的其他字段来展开该项。
- EditItemTemplate：指定当某项处于编辑模式中时的布局。此模板通常包含一些编辑控件，如 TextBox 控件。
- HeaderTemplate 和 FooterTemplate：包含在列表的开始和结束处分别呈现的文本和控件。
- SeparatorTemplate：包含在每项之间呈现的元素。典型的示例可能是一条直线(使用 HR 元素)。

设计者需要根据不同的需要定义不同类型的项模板，DataList 控件根据项的运行时状态自动加载相应的模板显示数据，例如当某一项被选定后将会以SelectedItemTemplate模板呈现数据，编辑功能被激活时将以EditItemTemplate模板呈现数据。

下面以【例 8-7】来说明如何通过设置模板为 DataList 控件定义数据的呈现样式并完成数据绑定，

【例 8-7】DataList 控件的数据绑定。

(1) 在 DataBinding 网站中新建一个名为 DataListBingding.aspx 的页面，在页面上添加一个 DataList 控件。

(2) 为 DataList 编辑、设置项模板并进行显示字段影射。

在 VWD 2008 环境中使用 DataList 控件的快捷任务面板进入模板的编辑页面，如图 8-27 所示。



图 8-27　打开 DataList 的模板编辑器

单击编辑模板按钮后进入模板编辑界面，如图 8-28 所示。

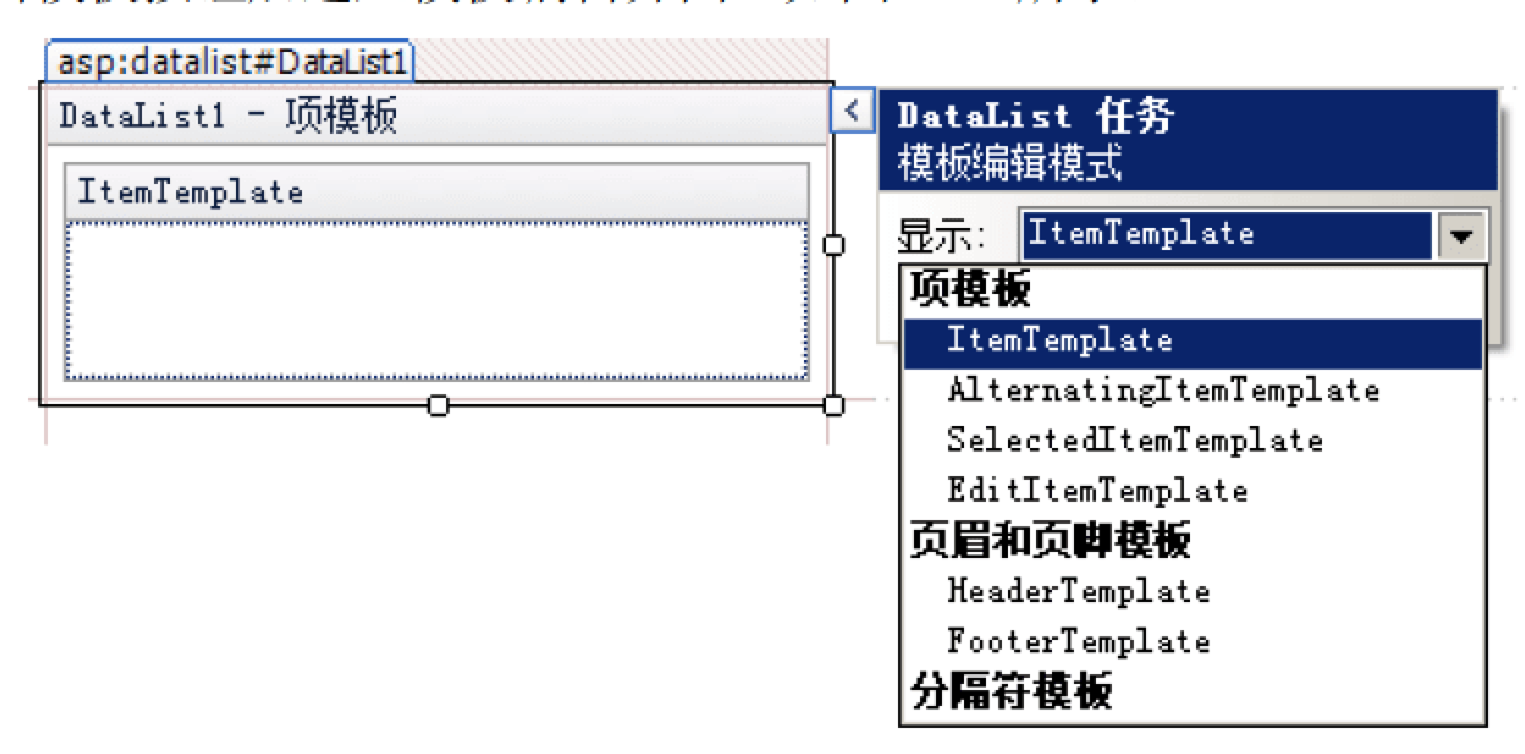


图 8-28　模板编辑界面

在这里只实现 DataList 控件的数据绑定，所以只简单地定义一个 ItemTemplate，单击模板类型后编辑 ItemTemplate 模板样式如图 8-29 所示。

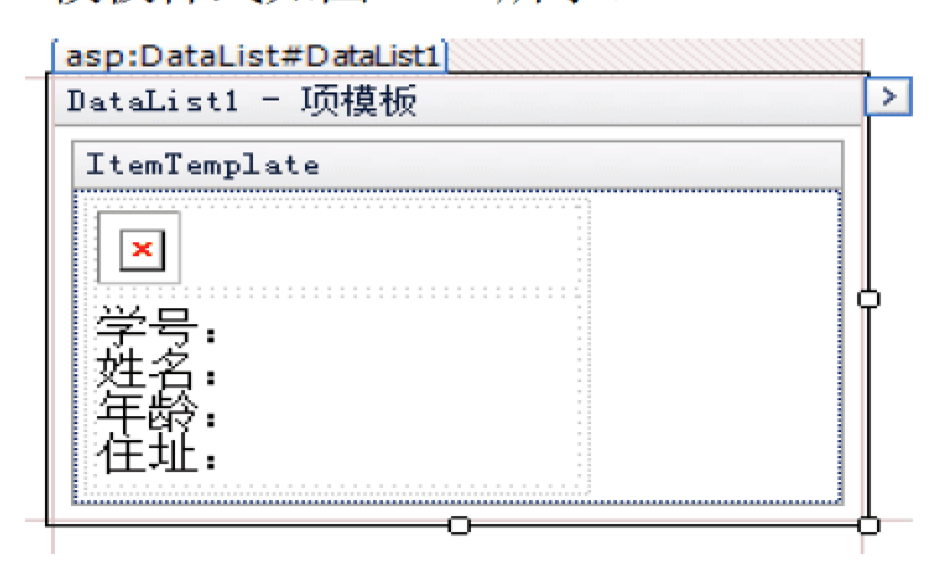


图 8-29　ItemTemplate 模板样式

ItemTemplate 模板样式中，包含一个 2 行 1 列的 HTML Table，第 1 行显示图片，第 2 行显示记录中的其他字段。回顾 GridView 控件中，设置绑定列时需要同时设置绑定列到数据字段之间的数据映射，DataList 控件中的项模板显示数据源每条记录中的各个字段，也需要将模板中的显示控件影射到相应字段，才能在数据绑定后在模板项中显示正确的数据。数据影射通过绑定表达式完成，在项模板中各个显示控件的页面代码中添加如下绑定表达式：<%# Eval("XXX") %>，其中 Eval 方法用于读取数据绑定后当前显示项中所呈现的数据项(某条记录)的相应字段数据，Eval 方法的参数 XXX 指定记录中要显示的字段名。可以这样来理解<%# Eval("XXX") %>表达式，当在后台代码中为某种数据绑定控件(如这里的 DataList)设置数据源

并进行数据绑定后，运行时数据源中的记录就会自动与显示项关联，有这种关联作为上下文，只要指定字段名就可以访问到该记录中的字段数据。因为 Eval 方法需要在数据上下文中读取数据，因此<%# Eval("XXX") %>表达式只能用在数据绑定控件的模板定义中。

定义模板后的页面代码如下：

```
<div>
    <asp:DataList ID="DataList1" runat="server" Height="354px" RepeatColumns="3"
        HorizontalAlign="Justify" RepeatDirection="Horizontal">
        <ItemTemplate>
            <table style="width: 154px; height: 111px">
                <tr>
                    <td style="width: 100px">
                        <img alt="照片" src='./image/<%# Eval("photo") %>' /></td>
                </tr>
                <tr>
                    <td style="width: 100px">
                        学号：<%# Eval("no") %><br />
                        姓名：<%# Eval("name") %><br />
                        年龄：<%# Eval("birth") %><br />
                        住址：<%# Eval("address") %></td>
                </tr>
            </table>
        </ItemTemplate>
    </asp:DataList>
</div>
```

项模板第 1 行图片控件中的表达式<%# Eval("photo") %>读取数据源记录中的 photo 字段值作为图片名称。

(3) 设置 DataList 的布局属性，采用 Table 布局，每行显示 5 个项，按行显示，如图 8-30 所示。

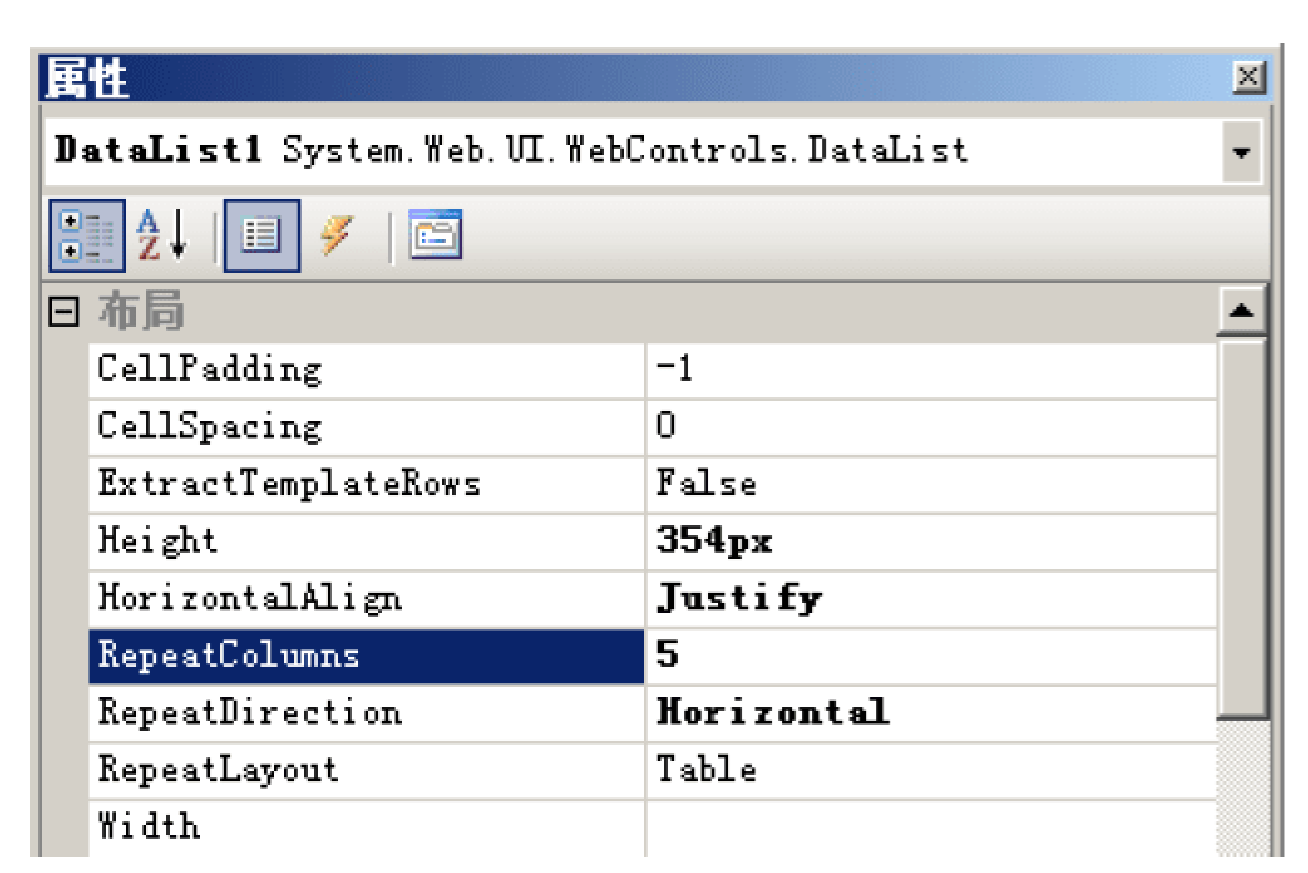


图 8-30　设置 DataList 布局属性

(4) 在页面后台类中添加数据绑定代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack) listbind();
}
void listbind()
{
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString; ;
    DataSet ds = new DataSet();
    using (SqlConnection sqlconn = new SqlConnection(sqlconnstr))
    {
        SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
        sqld.Fill(ds, "tabstudent");
    }
    //以数据集中名为tabstudent的DataTable作为数据源，为控件绑定数据
    DataList1.DataSource = ds.Tables["tabstudent"].DefaultView;
    DataList1.DataBind();
}
```

(5) 页面的运行效果如图 8-31 所示。



图 8-31　DataListBinding.aspx 页面运行效果

8.4.2　FormView 控件的数据呈现和处理

FormView 控件提供了内置的数据处理功能，只需绑定到支持这些功能的数据源控件，并进行配置，无需编写代码就可以实现对数据的分页和增删改功能。要使用 FormView 内置的增删改功能需要为更新操作提供 EditItemTemplate 和 InsertItemTemplate 模板，FormView 控件显示指定的模板以提供允许用户修改记录内容的用户界面。每个模板都包含用户可以单击以执行编辑或插入操作的命令按钮。用户单击命令按钮时，FormView 控件使用指定的编辑

或插入模板重新显示绑定记录以允许用户修改记录。插入或编辑模板通常包括一个允许用户显示空白记录的【插入】按钮或保存更改的【更新】按钮。用户单击【插入】或【更新】按钮时，FormView 控件将绑定值和主键信息传递给关联的数据源控件，该控件执行相应的更新。例如，SqlDataSource 控件使用更改后的数据作为参数值来执行 SQL Update 语句。

由于 FormView 控件的各个项通过自定义模板来呈现，因此控件并不提供内置的实现某一功能(如删除)的特殊按钮类型，而是通过按钮控件的 CommandName 属性与内置的命令相关联。FormView 提供如下命令类型(区分大小写)。

- Edit：引发此命令控件转换到编辑模式，并用已定义的 EditItemTemplate 呈现数据。
- New：引发此命令控件转换到插入模式，并用已定义的 InsertItemTemplate 呈现数据。
- Update：此命令将使用用户在 EditItemTemplate 界面中的输入值在数据源中更新当前所显示的记录。引发 ItemUpdating 和 ItemUpdated 事件。
- Insert：此命令用于将用户在 InsertItemTemplate 界面中的输入值在数据源中插入一条新的记录。引发 ItemInserting 和 ItemInserted 事件。
- Delete：此命令删除当前显示的记录。引发 ItemDeleting 和 ItemDeleted 事件。
- Cancel：在更新或插入操作中取消操作和放弃用户输入值，然后控件会自动转换到 DefaultMode 属性指定的模式。

在命令所引发的事件中，可以执行一些额外的操作，例如对于 Update 和 Insert 命令，因为 ItemUpdating 和 ItemInserting 事件是在更新或插入数据源之前触发的，可以在 ItemUpdating 和 ItemInserting 事件中先判断用户的输入值进行验证，满足要求后才访问数据库，否则取消操作。

下面通过【例 8-8】演示如何使用 FormView 控件完成数据的分页呈现，编辑、更新、删除、添加等数据处理功能。

【例 8-8】在 FormView 控件中实现数据的分页呈现、编辑、更新、删除和添加操作。

(1) 在 DataBinding 网站中新建一个名为 FormViewBinding.aspx 的页面，在页面上添加 1 个 FormView 控件。

(2) 为 FormView 控件添加并编辑项模板，由于要实现数据的更新和插入操作，需要 3 种项模板 ItemTemplate、EditItemTemplate 和 InsertItemTemplate，分别在显示、更新和插入状态下呈现数据。在 FormView 控件中也提供模板编辑界面(如 DataList 控件)，这里直接在页面代码中进行编辑，首先编辑 EditItemTemplate 页面代码如下：

```
<EditItemTemplate>
    <table style="width:100%;">
        <tr>
            <td width="40%"><asp:Label ID="Label1" runat="server" Text="学号"
                                        Width="100%"></asp:Label> </td>
             <td width="60%"><asp:Label ID="noLabel1" runat="server" Text='<%# Eval("no") %>' /></td>
```

```
            </tr>
            <tr>
                <td width="40%"><asp:Label ID="Label2" runat="server" Text="姓名"
                                        Width="100%"></asp:Label> </td>
                <td width="60%"><asp:TextBox ID="nameTextBox" runat="server" Text='<%# Bind("name") %>'
                                        /></td>
            </tr>
            <tr>
                 <td width="40%"><asp:Label ID="Label3" runat="server" Text="出生日期"
                                        Width="100%"></asp:Label> </td>
                 <td width="60%"> <asp:TextBox ID="birthTextBox" runat="server" Text='<%# Bind("birth") %>'
                                        /></td>
             </tr>
                <tr>
                    <td width="40%"><asp:Label ID="Label4" runat="server" Text="地址"
                                        Width="100%"></asp:Label> </td>
                    <td width="60%"><asp:TextBox ID="addressTextBox" runat="server" Text='<%#
                                        Bind("address") %>' /></td>
                </tr>
                <tr>
                    <td width="40%"> </td>
                    <td width="60%" align="center">
                    <asp:LinkButton ID="UpdateButton" runat="server" CausesValidation="True"
                                        CommandName="Update" Text="更新" />
                    <asp:LinkButton ID="UpdateCancelButton" runat="server" CausesValidation="False"
                                        CommandName="Cancel" Text="取消" />
                      </td>
                 </tr>
            </table>
</EditItemTemplate>
```

编辑状态模板中用 1 个 Label 控件和 1 个 TextBox 控件代表数据源中的一个字段，其中 TextBox 控件为绑定字段，共 4 行，对应于数据源记录中的 4 个字段。与 DataList 中不同，这里 TextBox 控件的绑定表达式为<%# Bind("address") %>，表达式标记中调用了 Bind 方法(DataList 中使用 Eval 方法)，Bind 方法构成与数据源的双向影射，通过双向影射配合数据源控件可以完成控件内置的更新操作(仅对数据源控件有效)。而 DataList 控件中使用的 Eval 方法为单向影射不能更新数据。

用于数据显示的 ItemTemplate 和用于插入的 InsertItemTemplate 与之类似，这里不再赘述。

(3) 配置完成各个项模板后，为 FormView 配置分页，由于分页功能是内置的，只需要设置 FormView 控件的分页属性，如图 8-32 所示。

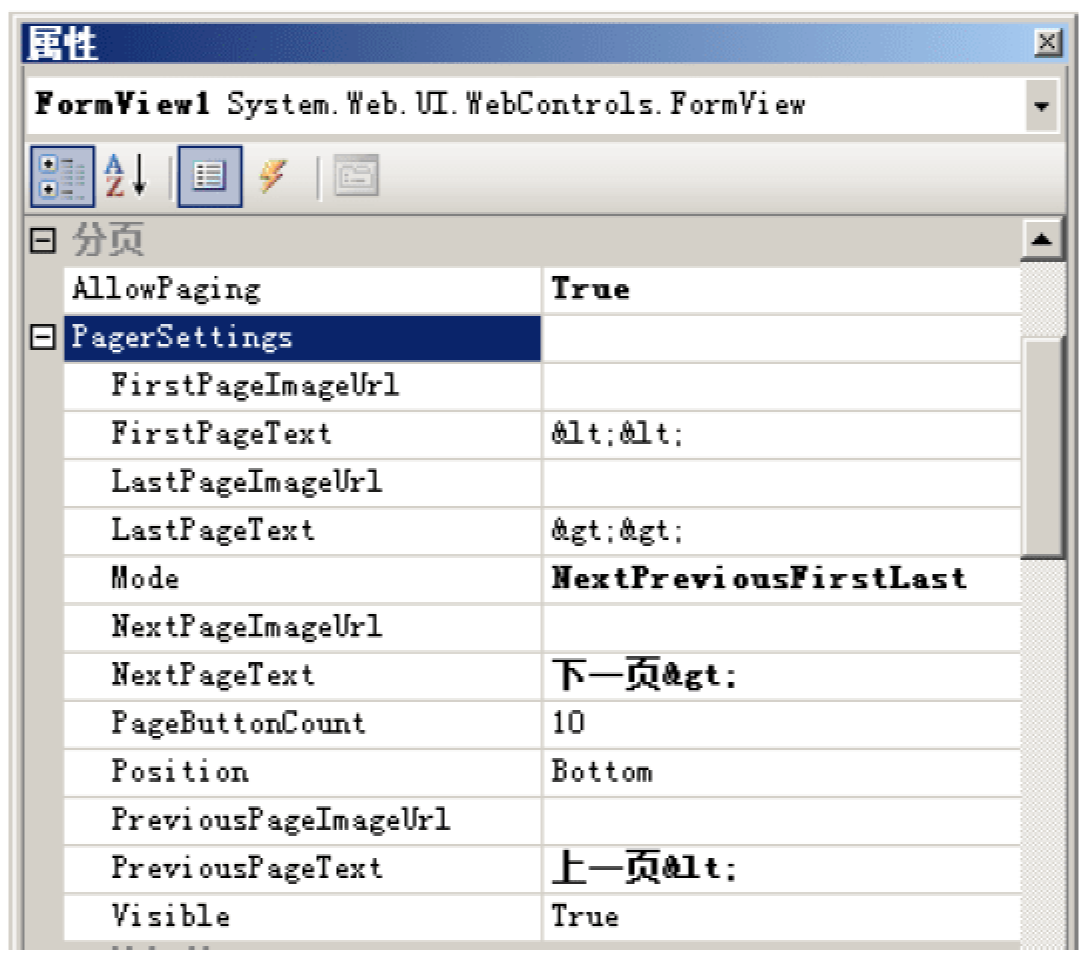


图 8-32　为 FormView 设置分页属性

(4) 调整 FormView 控件的外观，设置页眉模板 HeaderTemplate 完成整个页面的设计，完整的 FormView 控件设计代码如下：

```
<asp:FormView ID="FormView1" runat="server" DataKeyNames="no"
DataSourceID="SqlDataSource1" AllowPaging="True" CellPadding="4"    ForeColor="#333333" Width="231px">
<PagerSettings Mode="NextPreviousFirstLast" NextPageText="下一页&amp;gt;"
PreviousPageText="上一页&amp;lt;" />
<FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
<RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
<EditItemTemplate>
  <table style="width:100%;">
    <tr>
      <td width="40%"><asp:Label ID="Label1" runat="server" Text="学号" Width="100%"></asp:Label> </td>
      <td width="60%"><asp:Label ID="noLabel1" runat="server" Text='<%# Eval("no") %>' /></td>
   </tr>
   <tr>
      <td width="40%"><asp:Label ID="Label2" runat="server" Text="姓名" Width="100%"></asp:Label> </td>
      <td width="60%"><asp:TextBox ID="nameTextBox" runat="server" Text='<%# Bind("name") %>' /></td>
   </tr>
   <tr>
       <td width="40%"><asp:Label ID="Label3" runat="server" Text="出生日期" Width="100%"></asp:Label>
      </td>
      <td width="60%"> <asp:TextBox ID="birthTextBox" runat="server" Text='<%# Bind("birth") %>' /></td>
   </tr>
   <tr>
      <td width="40%"><asp:Label ID="Label4" runat="server" Text="地址" Width="100%"></asp:Label> </td>
      <td width="60%"><asp:TextBox ID="addressTextBox" runat="server" Text='<%# Bind("address") %>'
       /></td>
   </tr>
   <tr>
```

```
            <td width="40%"> </td>
            <td width="60%" align="center">
              <asp:LinkButton ID="UpdateButton" runat="server" CausesValidation="True"
              CommandName="Update" Text="更新" />
             <asp:LinkButton ID="UpdateCancelButton" runat="server" CausesValidation="False"
              CommandName="Cancel" Text="取消" />
            </td>
    </tr>
       </table>
        </EditItemTemplate>
        <InsertItemTemplate>
            <table style="width:100%;">
              <tr>
                <td width="40%"><asp:Label ID="Label2" runat="server" Text="姓名" Width="100%"></asp:Label>
                </td>
                <td width="60%"><asp:TextBox ID="nameTextBox" runat="server" Text='<%# Bind("name") %>'
                /></td>
             </tr>
             <tr>
                <td width="40%"><asp:Label ID="Label3" runat="server" Text="出生日期"
                 Width="100%"></asp:Label> </td>
                <td width="60%"><asp:TextBox ID="birthTextBox" runat="server" Text='<%# Bind("birth") %>'
                 /></td>
            </tr>
            <tr>
                 <td width="40%"><asp:Label ID="Label4" runat="server" Text="地址"
                   Width="100%"></asp:Label> </td>
                 <td width="60%"><asp:TextBox ID="addressTextBox" runat="server"
                   Text='<%# Bind("address") %>' /></td>
           </tr>
           <tr>
                 <td width="40%"> </td>
                 <td width="60%" align="center">
                   <asp:LinkButton ID="InsertButton" runat="server" CausesValidation="True"
                     CommandName="Insert" Text="插入" />
                   <asp:LinkButton ID="InsertCancelButton" runat="server" CausesValidation="False"
                     CommandName="Cancel" Text="取消" />
                 </td>
           </tr>
              </table>
              </InsertItemTemplate>
              <ItemTemplate>
                <table style="width:100%;">
                   <tr>
```

```
                <td width="40%"><asp:Label ID="Label1" runat="server" Text="学号"
                 Width="100%"></asp:Label> </td>
                <td width="60%"><asp:Label ID="noLabel" runat="server" Text='<%# Eval("no") %>' /></td>
            </tr>
            <tr>
                <td width="40%"><asp:Label ID="Label2" runat="server" Text="姓名"
                 Width="100%"></asp:Label> </td>
                <td width="60%"><asp:Label ID="nameLabel" runat="server" Text='<%# Bind("name") %>'
                  /></td>
            </tr>
            <tr>
                <td width="40%"><asp:Label ID="Label3" runat="server" Text="出生日期"
                 Width="100%"></asp:Label> </td>
                <td width="60%"><asp:Label ID="birthLabel" runat="server" Text='<%# Bind("birth") %>'
                  /></td>
            </tr>
            <tr>
                 <td width="40%"><asp:Label ID="Label4" runat="server" Text="地址"
                   Width="100%"></asp:Label> </td>
                <td width="60%"><asp:Label ID="addressLabel" runat="server"
                  Text='<%# Bind("address") %>' /></td>
           </tr>
           <tr>
               <td width="40%"> </td>
               <td width="60%" align="right">
                  <asp:LinkButton ID="NewButton" runat="server" CausesValidation="True"
                    CommandName="New" Text="新建" />
                  <asp:LinkButton ID="EditButton" runat="server" CausesValidation="False"
                    CommandName="Edit" Text="更新" />
                  <asp:LinkButton ID="DeleteButton" runat="server" CausesValidation="False"
                    CommandName="Delete" Text="删除" />
               </td>
           </tr>
        </table>
    </ItemTemplate>
<PagerStyle BackColor="#FFCC66" ForeColor="#333333" HorizontalAlign="Center" />
<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
     <HeaderTemplate>
          学生详细信息
     </HeaderTemplate>
</asp:FormView>
```

页面设计外观如图 8-33 所示。



图 8-33　FormViewBinding 页面设计效果

页面设计完成后为控件创建并配置数据源控件，需要在数据源控件中配置用于实现增删改查功能的查询语句以支持 FormView 控件中的相应命令按钮。

(5) 在页面上添加一个 SqlDataSource 数据源控件，过程如下：

首先为数据源控件配置数据库连接，依然使用第 7 章介绍过的 mydatabase.mdf。如图 8-34 所示。

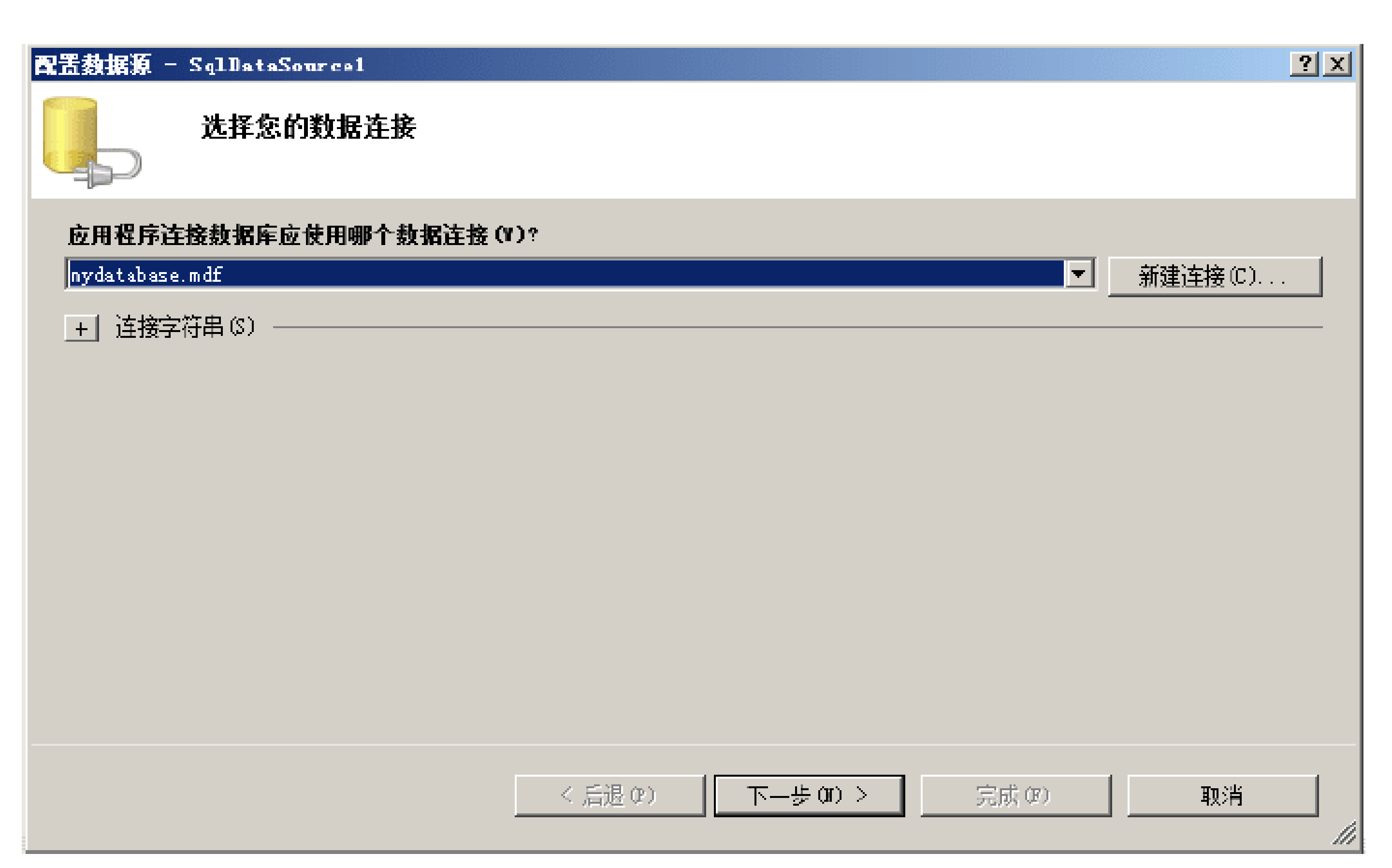


图 8-34　为数据源控件配置连接

单击下一步配置连接字符串，系统将自动生成 mydatabose.mdf 的连接字符串并保存，如果已经按照第 7 章给出的手动方式将连接字符串添加到 web.config 文件中，则可以跳过此步。

然后为数据源配置查询如图 8-35 所示。

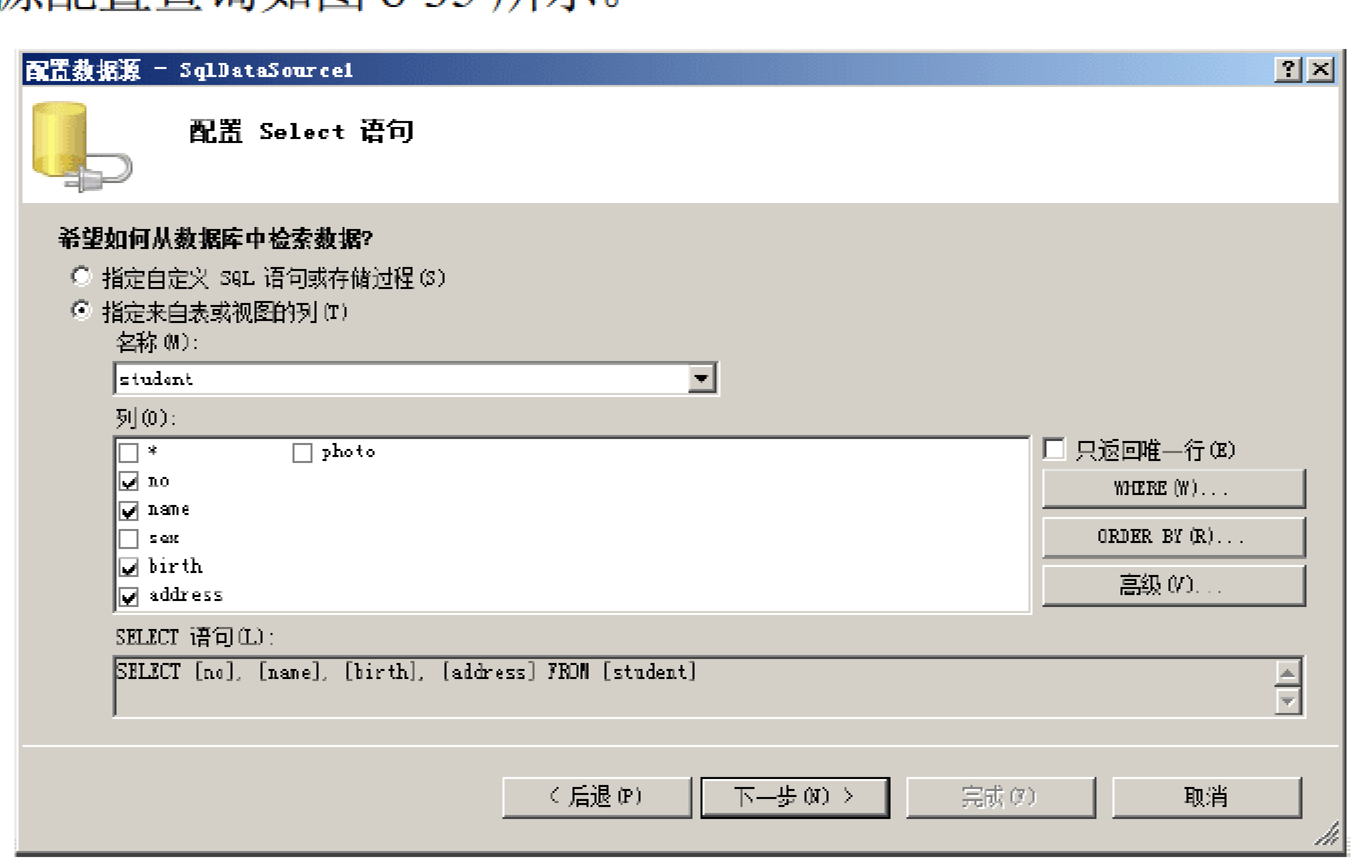


图 8-35　配置查询

完成后为 SqlDataSource 数据源控件配置更新、插入和删除语句。

首先打开数据源控件的属性窗口在其数据属性部分设置 DeleteCommandType 属性，该属性支持 2 种枚举值，Text 和 StoredProcedure，前者使用 Sql 语句实现删除操作，后者用数据库中的存储过程实现，例中设置为 Text。

如图 8-36 所示。

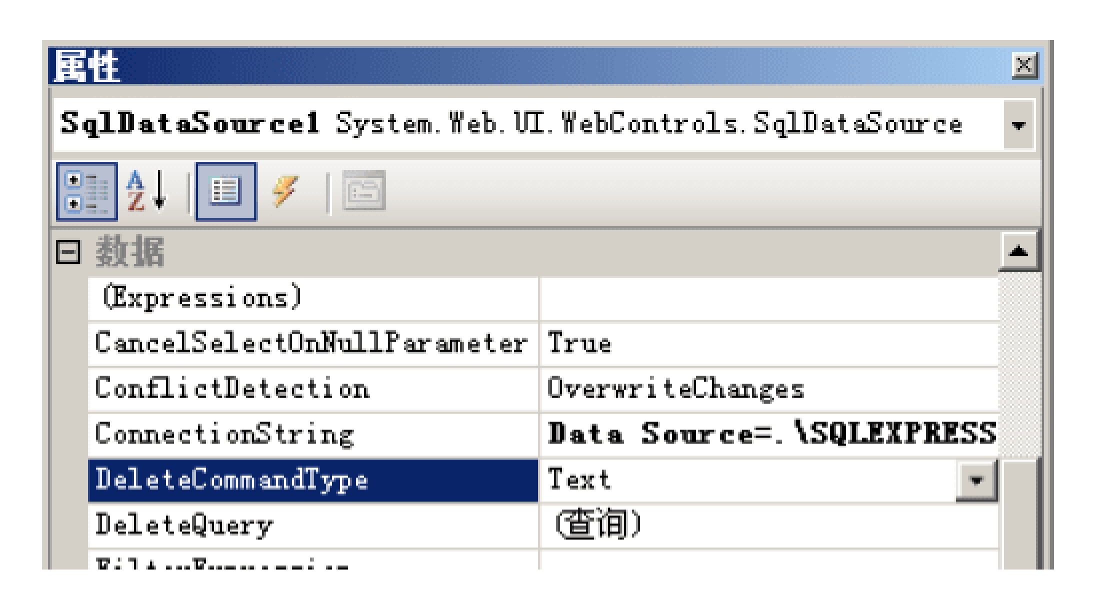


图 8-36　为 SqlDataSource 控件配置属性

然后为查询操作添加 Sql 语句完成删除操作，通过 DeleteQuery 属性打开删除命令的【命令和参数编辑器】界面，如图 8-37 所示。



图 8-37　【命令和参数编辑器】界面

在 DELETE 命令一栏输入删除语句：DELETE FROM student WHERE (no = @no)。这里需要注意 Delete 语句的参数 no = @no，正因为在编辑模板中为数据显示项和数据源记录字段之间建立了双向绑定，才可以直接将绑定字段名 no 作为参数名完成删除语句，系统将自动获取编辑模板中的相应显示项的当前显示值作(对于插入和更新操作是输入值)为参数执行 Sql 语句。

用同样的方法为更新操作添加 Update 语句：

```
UPDATE student SETname = @name,birth=@birth,address=@address where no=@no
```

为插入操作添加 Insert 语句：

```
INSERT INTO student(name, birth, address) VALUES (@name,@birth,@address)
```

上述设置将会在页面前台生成如下 SqlDataSource 控件定义代码：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\mydatabase.mdf;Integrated Security=True;User
Instance=True"
            ProviderName="System.Data.SqlClient"
            SelectCommand="SELECT [no], [name], [birth], [address] FROM [student]"

            InsertCommand="INSERT INTO student(name, birth, address) VALUES
(@name,@birth,@address)"
            DeleteCommand="DELETE FROM student WHERE (no = @no)"
            UpdateCommand="update student set name = @name,birth=@birth,address=@address where
no=@no">
</asp:SqlDataSource>
```

(6) 到此已全部完成了 FormView 控件的设计，可以运行查看执行的效果，如图 8-38 和图 8-39 所示。

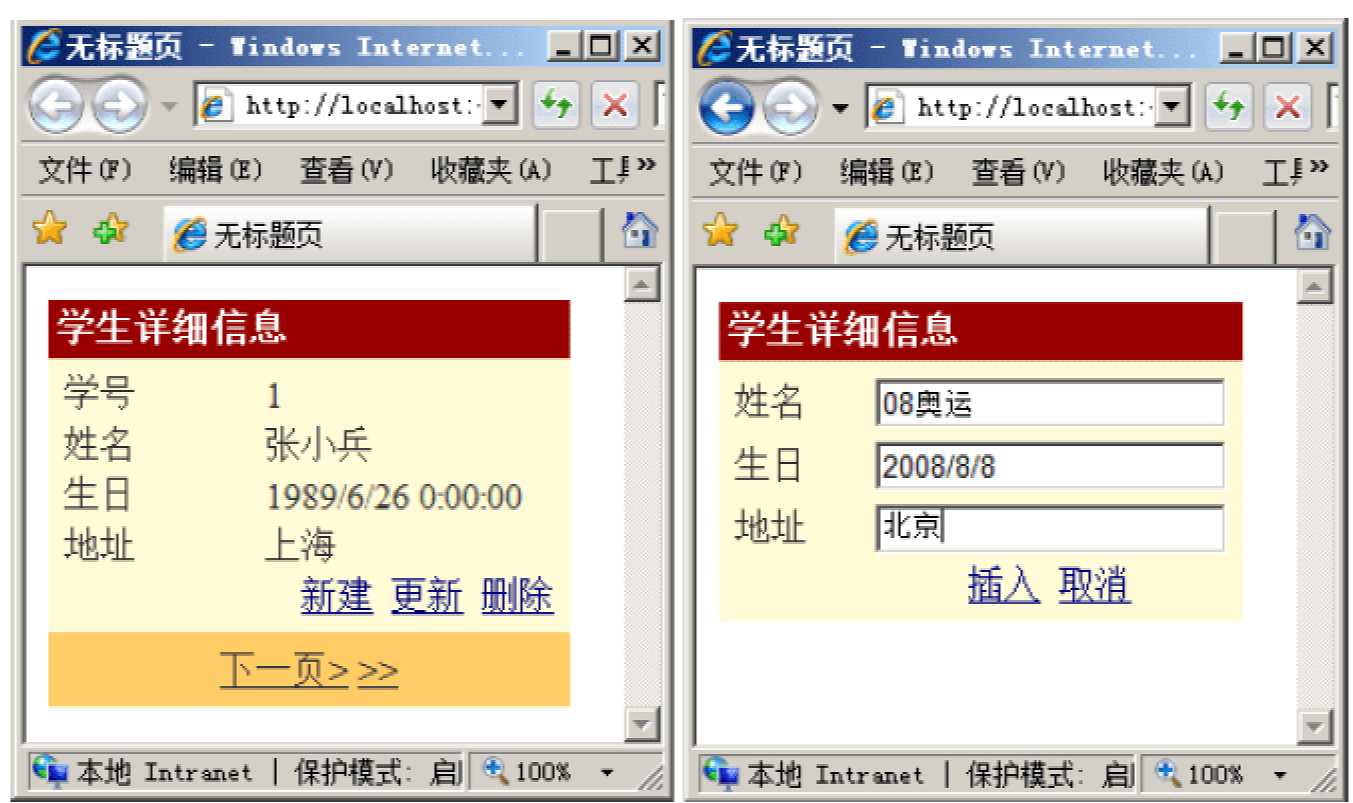


图 8-38　FormViewBinding.aspx 的显示和插入界面



图 8-39　FormViewBinding.aspx 的显示和更新界面

实验 8　数据绑定

1. 实验目的

熟悉 ASP.NET 中的数据绑定，掌握使用 GridView、DataList 和 FormView 进行数据显示和处理的方法。

2. 实验内容和要求

(1) 新建名字为 DataBinding_ Exercise 的网站。

(2) 在网站中建立用于数据绑定的数据库(可参考本章使用的实例数据库 student，或直接使用第 7 章实验中的 MyDatabase_ Exercis)。

(3) 添加一个网页，利用 GridView 实现数据的分页显示。

(4) 添加一个网页，利用 DataList 实现数据的分页显示。

(5) 添加一个网页，利用 FormView 控件实现数据的插入、修改和删除操作，FormView 界面及布局自定义。

第9章　ASP.NET AJAX

ASP.NET AJAX 采用异步编程方式，与前面学习的同步编程模式有所不同，最大的特点是提供对客户端脚本的自动管理，利用 ASP.NET AJAX 服务器控件，程序员可以使用 ASP.NET 3.5 中现有的各种控件实现局部页面更新的效果。

本章首先介绍 ASP.NET AJAX 的基本知识，然后详细讲解 ASP.NET AJAX 主要服务器控件的使用方法，最后介绍 ASP.NET AJAX Control Extenders 扩展控件。

本章的学习目标

- 了解 ASP.NET AJAX 的基本知识
- 掌握 ASP.NET AJAX 主要控件的用法
- 了解 ASP.NET AJAX Control Extenders 扩展控件

9.1　ASP.NET AJAX 概述

Ajax(Asynchronous JavaScript and XML)技术是由 Jesse James Garrett 提出的，是综合异步通信、JavaScript 以及 XML 等多种网络技术新的编程方式。如果从用户看到的实际效果来看，也可以形象地称之为无页面刷新技术。

Ajax 技术的主要内容包括：基于 Web 标准 XHTML+CSS 的表示；使用 DOM(Document Object Model)进行动态显示及交互；使用 XML 和 XSLT 进行数据交换及相关操作；使用 XMLHttpRequest 进行异步数据检索；使用 JavaScript 将所有的东西绑定在一起等等。

Ajax 技术的最大优点就是能在不更新整个页面的前提下维护数据。这使得 Web 应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

2005 年，Microsoft 公司在专业开发人员大会上宣布将在 ASP.NET 上实现 Ajax 功能(开发代号为 Atlas)，主要是为了充分利用客户端 JavaScript、DHTML 和 XMLHttpRequest 对象。目的是帮助开发人员创建更具交互性的支持 Ajax 的 Web 应用程序。直到 2007 年 1 月，Microsoft 公司才真正推出了具有 Ajax 风格的方便的异步编程模型，这就是 ASP.NET AJAX 1.0。同时为了与其他的 Ajax 技术区分，Microsoft 公司用大写的 AJAX，并在其前面加上

ASP.NET。

ASP.NET AJAX 1.0 是以可以在 ASP.NET 2.0 之上安装的单独一个下载的形式发布的。从. NET Framework 3.5 开始，所有这些特性都成为 ASP.NET 所固有的，这意味着在构建或部署应用时，不再需要下载和安装单独的 ASP.NET AJAX 安装文件。

当在 VWD 2008 中创建针对.NET Framework 3.5 的新 ASP.NET 网站时，VWD 2008 会自动在 web.config 文件里添加适当的 AJAX 注册设置，而且核心 ASP.NET AJAX 服务器控件会出现在工具箱里。

对于 Web 开发来说，ASP.NET AJAX 从基础框架实现，到客户端与服务器的通信，都发生了翻天覆地的变化。相对于 ASP.NET 来说，ASP.NET AJAX 是一种更为成熟的 Web 开发技术。

下面介绍 ASP.NET AJAX 主要控件 ScriptManager、UpdatePanel、UpdateProgress 和 Timer 的编程方法。

9.2　ScriptManager 控件

ScriptManager 控件是 ASP.NET AJAX 的核心，它提供处理页面上的所有 ASP.NET AJAX 控件(UpdatePanel、UpdateProgress 等)的支持，没有该控件的存在其他 ASP.NET AJAX 控件是不能工作的，并且所有需要支持 ASP.NET AJAX 的 ASP.NET 页面上只能有一个 ScriptManager 控件。另外，ScriptManager 控件还可以生成相关的客户端代理脚本以便能够在客户端脚本中访问 Web Service。

9.2.1　ScriptManager 控件的属性和方法

要使用 ASP.NET AJAX 提供的功能，必须在网页中包含一个 ScriptManager 控件。如果在网站中添加一个【AJAX Web 窗体】网页，ASP.NET 会自动在该网页中添加一个 ScriptManager 控件标记，代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
```

但是，如果在网站中添加一个【Web 窗体】网页，ASP.NET 就不会自动添加这个控件，此时需要从【工具箱】中将其拖放到页面中，使其生成下面代码：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

ScriptManager 控件提供了很多属性和方法，用于对客户端脚本进行各种复杂的管理。表 9-1 列出了部分内容。

表 9-1　ScriptManager 控件的常用属性和方法

属 性 方 法	说　　明
AsyncPostBackErrorMessage 属性	异步回传发生错误时的自定义提示错误信息
AsyncPostBackTimeout 属性	异步回传时超时限制，默认值为 90，单位为秒
EnablePartialRendering 属性	是否支持页面的局部更新，默认值为 True，一般不需要修改
ScriptMode 属性	指定 ScriptManager 发送到客户端的脚本的模式，有 4 种模式：Auto，Inherit，Debug，Release，默认值为 Auto
ScriptPath 属性	设置所有的脚本块的根目录，作为全局属性，包括自定义的脚本块或者引用第三方的脚本块。如果在 Scripts 中的 <asp:ScriptReference />标签中设置了 Path 属性，它将覆盖该属性
OnAsyncPostBackError 方法	异步回传发生异常时的服务端处理函数，在这里可以捕获一场信息并作相应的处理
OnResolveScriptReference 方法	指定 ResolveScriptReference 事件的服务器端处理函数，在该函数中可以修改某一条脚本的相关信息如路径、版本等

9.2.2　ScriptManager 控件的用法

下面是一个简单的 ASP.NET AJAX 示例，实现将 DropDownList 已选择值，传递给一个 Label 控件的 Text 属性。

【例 9-1】简单的 ASP.NET AJAX 示例。

(1) 创建名称为 ScriptManagerExample 的网站。

(2) 在 ScriptManagerExample 网站中添加一个名为 Example.aspx 的【Web 窗体】网页。

(3) 将 2 个控件 ScriptManager 和 UpdatePanel 拖放到 Example.aspx 的【设计】视图中，然后在 UpdatePanel 中放入 DropDownList 和 Label 2 个控件。

(4) 给 DropDownList 控件增加 2 个选择项 hello world 和 hello ajax，同时设置 DropDownList 控件的 AutoPastBack 属性为 True。

网页【设计】视图如图 9-1 所示。

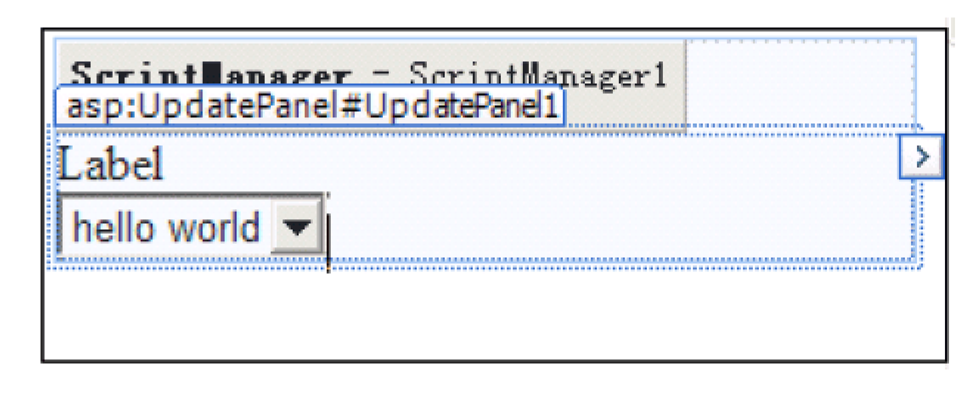


图 9-1　Example.aspx 的设计视图

对应【源】视图中的部分代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <br />
        <asp:DropDownList ID="DropDownList1" runat="server"
                onselectedindexchanged="DropDownList1_SelectedIndexChanged"
                 AutoPostBack="True" >
                <asp:ListItem>hello world</asp:ListItem>
                <asp:ListItem>hello ajax</asp:ListItem>
        </asp:DropDownList>
    </ContentTemplate>
</asp:UpdatePanel>
```

(5) 双击【设计】视图中的 DropDownList 控件，添加下面后台代码。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
        Label1.Text = DropDownList1.SelectedValue;
}
```

(6) 按 Ctrl＋F5 组合键运行，选择 DropDownList 控件的选项，观察页面刷新效果。此时可以看到，浏览器的【后退】按钮是灰色的，该示例不像其他程序那样，出现屏幕“闪烁”，说明它没有进行整个页面的更新。

9.3　UpdatePanel 控件

UpdatePanel 控件是 ASP.NET AJAX 中很重要的一个控件，它可以用来创建局部更新的 Web 应用程序。有了 UpdatePanel 控件，开发者不需要编写任何客户端脚本，只需在页面上添加 UpdatePanel 控件和 ScriptManager 控件就可以自动实现局部更新。

9.3.1　UpdatePanel 控件工作原理

UpdatePanel 控件的工作依赖于 ScriptManager 控件和客户端 PageRequestManager 类(Sys.WebForms.PageRequestManager，可以参考其他相关资料)，当 ScriptManager 允许页面局部更新时，它会以异步的方式回传给服务器，与传统的整页回传方式不同的是只有包含在 UpdatePanel 中的页面部分会被更新，在从服务器返回 XHTML 之后，PageRequestManager 会通过操作 DOM 对象来替换需要更新的代码片段。UpdatePanel 工作过程如图 9-2 所示。

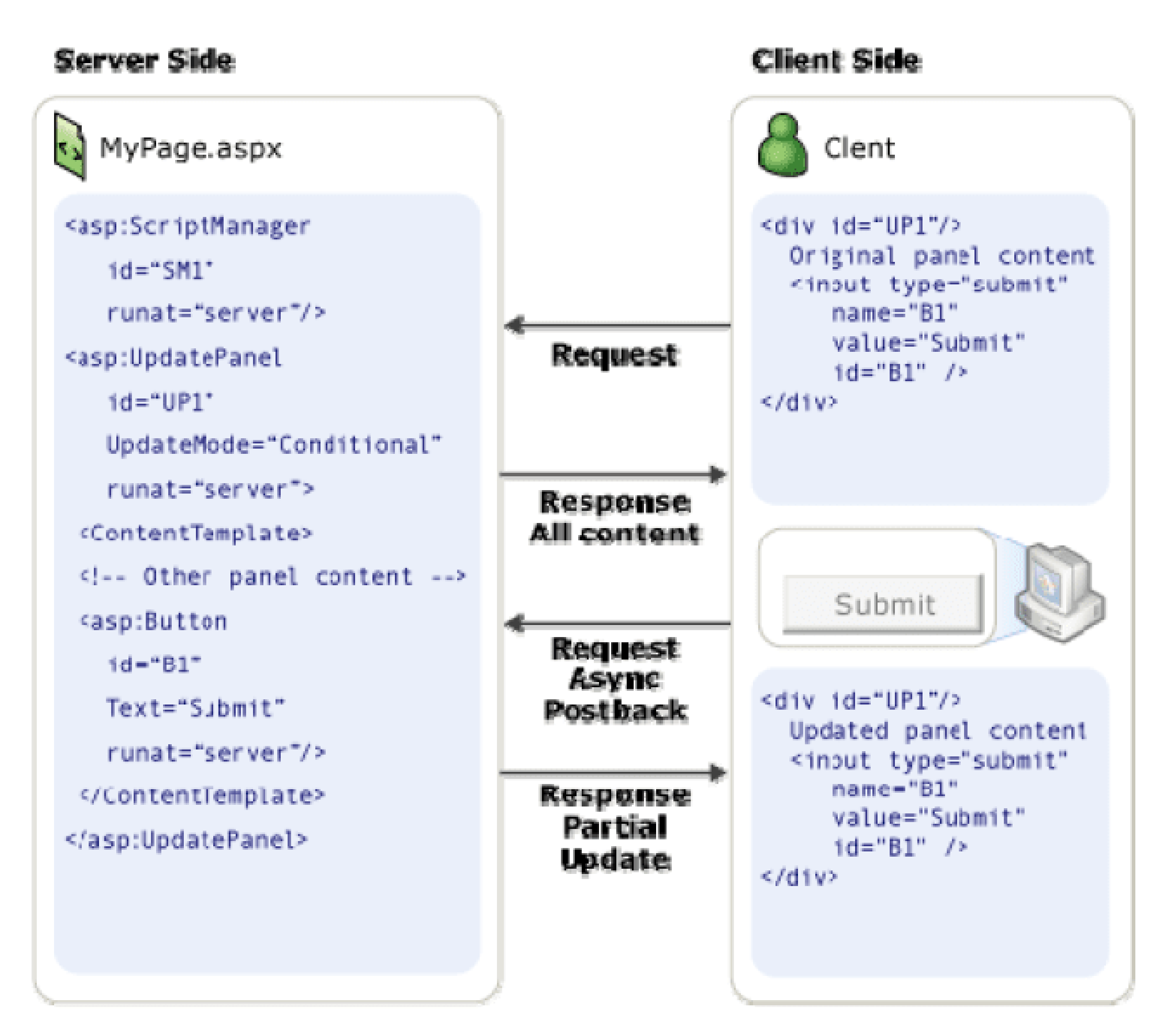


图 9-2　UpdatePanel 工作过程

从图 9-2 中可以看出，当客户端向服务器第 1 次发出请求时，服务器返回整个页面。除此之外，均通过异步回传方式对页面进行局部更新。

9.3.2　UpdatePanel 控件属性

UpdatePanel 控件常用属性如表 9-2 所示。

表 9-2　UpdatePanel 控件常用属性

属　　性	说　　明
ContenteTemplate	定义 UpdatePanel 的内容
Triggers	定义控件的服务器事件为异步或同步触发器
ChildrenAsTriggers	当 UpdateMode 属性为 Conditional 时，UpdatePanel 中的子控件的异步回送是否会引发 UpdatePanle 的更新
RenderMode	表示 UpdatePanel 最终呈现的 HTML 元素。Block(默认)表示<div>，Inline 表示<span>
UpdateMode	表示 UpdatePanel 的更新模式，有 2 个选项：Always 和 Conditional。Always 是不管有没有 Trigger，其他控件都将更新该 UpdatePanel，Conditional 表示只有当前 UpdatePanel 的 Trigger，或 ChildrenAsTriggers 属性为 true 时当前 UpdatePanel 中控件引发的异步回送或者整页回送，或是服务器端调用 Update()方法才会引发更新该 UpdatePanel

ContenteTemplate 和 Triggers 都是 UpdatePanel 控件的重要属性。

1. ContentTemplate 属性

ContenteTemplate 属性用来定义 UpdatePanel 的内容，其内容包括其他控件或 HTML 元

素。例如，在 UpdatePanel 的内容放置 1 个 Label 控件和 1 个 Image 控件，代码如下：

```
•••
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Label"> </asp:Label>
        <asp:Image ID="Image1" runat="server" />
    </ContentTemplate>
</asp:UpdatePanel>
•••
```

2. Triggers 属性

在 UpdatePanel 中有 2 种 Triggers，分别为 AsyncPostBackTrigger 和 PostBackTrigger。AsyncPostBackTrigger 用来指定某个服务器控件作为该 UpdatePanel 的异步更新触发器，PostBackTrigger 用来指定在 UpdatePanel 中的某个服务器控件作为同步更新触发器。异步更新触发器所引发的回传叫异步回传，引发页面局部更新。同步更新触发器所引发的回传叫同步回传，引发传统的整页回传。

例如，Button1 采用异步回传方式，而 Button2 采用同步回传方式，代码如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      •••
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="Button1"   />
        <asp:PostBackTrigger ControlID="Button2" />
    </Triggers>
</asp:UpdatePanel>
```

9.3.3　在 UpdatePanel 中实现局部更新

在一个页面中，如果需要局部更新的内容较少，可以放置 1 个 UpdatePanel 控件，在该控件内实现局部更新的效果。

1. 在 UpdatePanel 中实现局部更新

下面通过具体示例介绍在 UpdatePanel 中实现局部更新的方法。

【例 9-2】在 UpdatePanel 中实现局部更新。

(1) 创建名称为 UpdatePanelExample1 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置 ScriptManager，UpdatePanel 两个控件。

(3) 在 UpdatePanel 内部放入 2 个 Label 控件和 1 个 Button 控件，同时在 UpdatePanel 外面也放置 1 个 Button 控件。

该页面【设计】视图如图 9-3 所示。

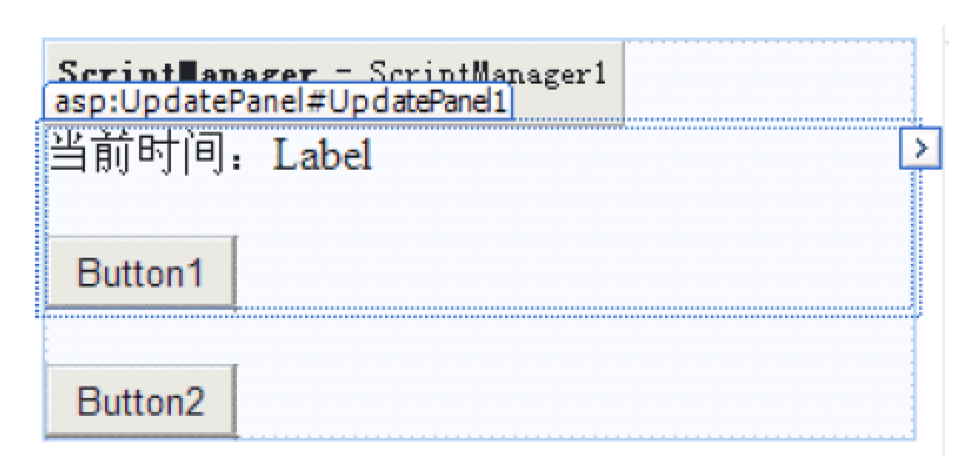


图 9-3　Default.aspx 的设计页面

对应【源】视图中的代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="当前时间："></asp:Label>
        <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
         <br />
        <asp:Button ID="Button1" runat="server" Text="Button1" onclick="Button1_Click" />
         <br />
    </ContentTemplate>
</asp:UpdatePanel>
<br />
<asp:Button ID="Button2" runat="server" Text="Button2"    onclick="Button2_Click" />
```

(4) 分别双击【设计】视图中 Button 控件，添加如下所示的后台代码。

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.Label2.Text = DateTime.Now.ToString();
}
protected void Button2_Click(object sender, EventArgs e)
{
    this.Label2.Text = DateTime.Now.ToString();
}
```

(5) 按 Ctrl＋F5 组合键运行，单击 Button1 和 Button2 按钮，观察有什么不同。

可以看到，虽然单击 Button1 和 Button2 按钮都实现了显示时间的 Label 控件更新，但页面刷新效果却不同。通常默认情况下，在 UpdatePanel 内部的服务器控件采用的是异步回传方式，实现 UpdatePanel 的局部更新，而在 UpdatePanel 外面的服务器控件采用的是同步回传方式，实现整个页面的刷新。

2. 利用 Triggers 属性指定服务器控件回传方式

下面通过对【例 9-2】简单的修改，介绍一下如何利用 Triggers 属性指定服务器控件回传方式。

【例 9-3】利用 Triggers 属性指定服务器控件回传方式。

(1) 打开【例 9-2】中 UpdatePanelExample1 网站。

(2) 在 Default.aspx 的【源】视图中增加 Triggers 标记，指定 Button1 控件为同步触发器，Button2 控件为异步触发器。代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="当前时间："></asp:Label>
        <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
        <br /> <br />
        <asp:Button ID="Button1" runat="server" Text="Button1" onclick="Button1_Click" />
        <br />
    </ContentTemplate>
    <Triggers>
          <asp:PostBackTrigger ControlID="Button1" />
          <asp:AsyncPostBackTrigger ControlID="Button2"   />
  </Triggers>
 </asp:UpdatePanel>
 <br />
 <asp:Button ID="Button2" runat="server" Text="Button2" onclick="Button2_Click" />
```

(3) 按 Ctrl＋F5 组合键运行。

此时可以看到，虽然 Button1 按钮在 UpdatePanel 内部，但实现的是整个页面的更新，而在 UpdatePanel 外面的 Button2 按钮却实现了 UpdatePanel 局部更新，究其原因是 Triggers 属性指定回传方式的结果。

9.3.4　同一页面上使用多个 UpdatePanel

使用 UpdatePanel 的时候并没有限制在一个页面上用多少个 UpdatePanel，所以可以为不同的需要局部更新的页面区域加上不同的 UpdatePanel。由于 UpdatePanel 默认的 UpdateMode 是 Always，如果页面上有一个局部更新被触发，则所有的 UpdatePanel 都将更新，这是有时不愿看到的，这时只需要 UpdatePanel 在它自己的触发器触发的时候更新就可以了，所以需要把 UpdateMode 设置为 Conditional。

下面的【例 9-4】包括 2 个 UpdatePanel，其中一个用来输入数据，而另一个则用来显示数据，每一个 UpdatePanel 的 UpdateMode 属性都设置为 Conditional，当单击【新增】按钮时，2 个 UpdatePanel 都更新，当单击【取消】按钮时，只有用来数据输入的那个 UpdatePanel 更新。

【例 9-4】同一页面上使用多个 UpdatePanel 的示例。

(1) 创建名称为 UpdatePanelExample3 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置 1 个 ScriptManager 和 2 个 UpdatePanel 控件。

(3) 在一个 UpdatePanel 中放入 1 个 ListBox 控件和 1 个 Label 控件，而在另一个

UpdatePanel 中放置 1 个 TextBox 控件、1 个 Label 控件和 2 个 Button 控件。

该页面【设计】视图如图 9-4 所示。

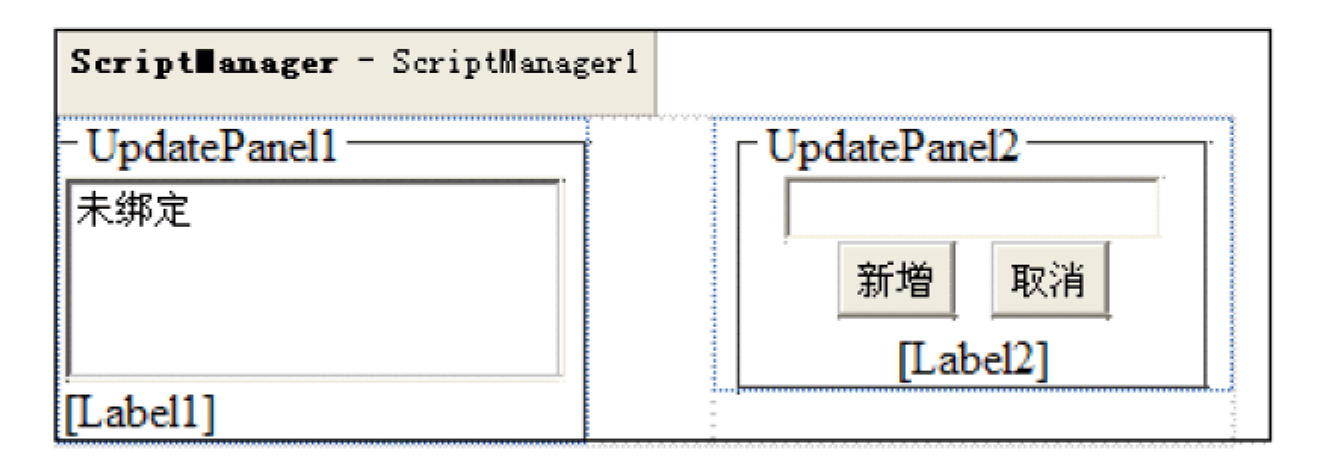


图 9-4　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
        •••
 <asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
      <ContentTemplate>
         <fieldset style="width:180px;">
         <legend >UpdatePanel1</legend>
         <asp:ListBox ID="ListBox1" runat="server"    Width="170px"></asp:ListBox>
         <asp:Label ID="Label1" runat="server" ><%=DateTime.Now %></asp:Label>
         </fieldset>
      </ContentTemplate>
      <Triggers>
          <asp:AsyncPostBackTrigger ControlID="InsertButton" EventName="Click" />
      </Triggers>
</asp:UpdatePanel>
      •••
<asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode="Conditional">
      <ContentTemplate>
         <fieldset style="width:160px;">
      <legend >UpdatePanel2</legend>
      <asp:TextBox ID="TextBox1" runat="server" EnableTheming="True"></asp:TextBox>
       <asp:Button ID="InsertButton" runat="server" Text="新增"    OnClick="InsertButton_Click" />
         
      <asp:Button ID="Cancelbutton" runat="server" Text="取消" OnClick="Cancelbutton_Click" />
       <asp:Label ID="Label2" runat="server" ><%=DateTime.Now %></asp:Label>
      </fieldset>
    </ContentTemplate>
  </asp:UpdatePanel>
```

(4) 分别双击【设计】视图中 Button 控件，添加如下所示的后台代码。

```
protected void InsertButton_Click(object sender, EventArgs e)
{
```

```
            ListBox1.Items.Add(TextBox1.Text);
            TextBox1.Text = String.Empty;
        }
        protected void Cancelbutton_Click(object sender, EventArgs e)
        {
            TextBox1.Text = String.Empty;
        }
```

(5) 按 Ctrl＋F5 组合键运行，程序运行效果如图 9-5 所示。

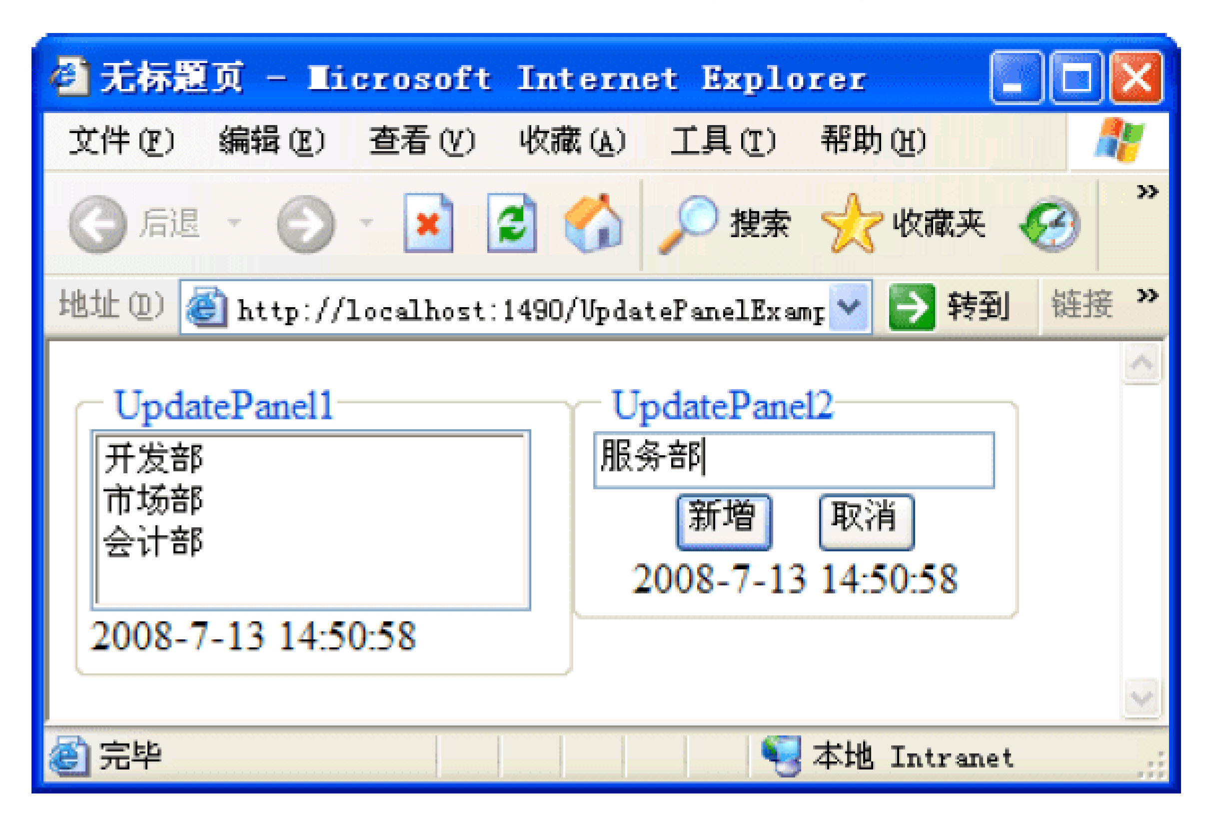


图 9-5　Default.aspx 的运行效果

此时可以看到，单击【新增】按钮同时更新 UpdatePanel1 和 UpdatePanel2，而单击【取消】按钮只会更新 UpdatePanel2。

【例 9-4】说明了如何在一个页面中使用多个 UpdatePanel，并将其 UpdateMode 设置为 Conditional 以及利用 Triggers 避免异步更新时多个 UpdatePanel 同时更新的情况。

9.3.5　在内容页中使用 UpdatePanel

由于网页发送到客户端时，母版页和内容页的内容是合并在一起的，并且网页只能有一个 ScriptManager 控件，因此只需要在母版页中放置 ScriptManager 控件，内容页就可以使用 UpdatePanel 了。

1. 在内容页中使用 UpdatePanel

下面通过具体示例介绍在内容页中使用 UpdatePanel 的方法。

【例 9-5】在内容页中使用 UpdatePanel 的示例。

(1) 创建名称为 UpdatePanelExample4 的网站。

(2) 在【解决方案资源管理器】中，添加一个名为 MasterPage.master 的母版页。

(3) 在工具箱的 AJAX Extensions 标签下双击 ScriptManager 控件，将 ScriptManager 控件添加到母版页中，确保 ScriptManager 控件添加在 ContentPlaceHolder 控件之外，如图 9-6 所示。

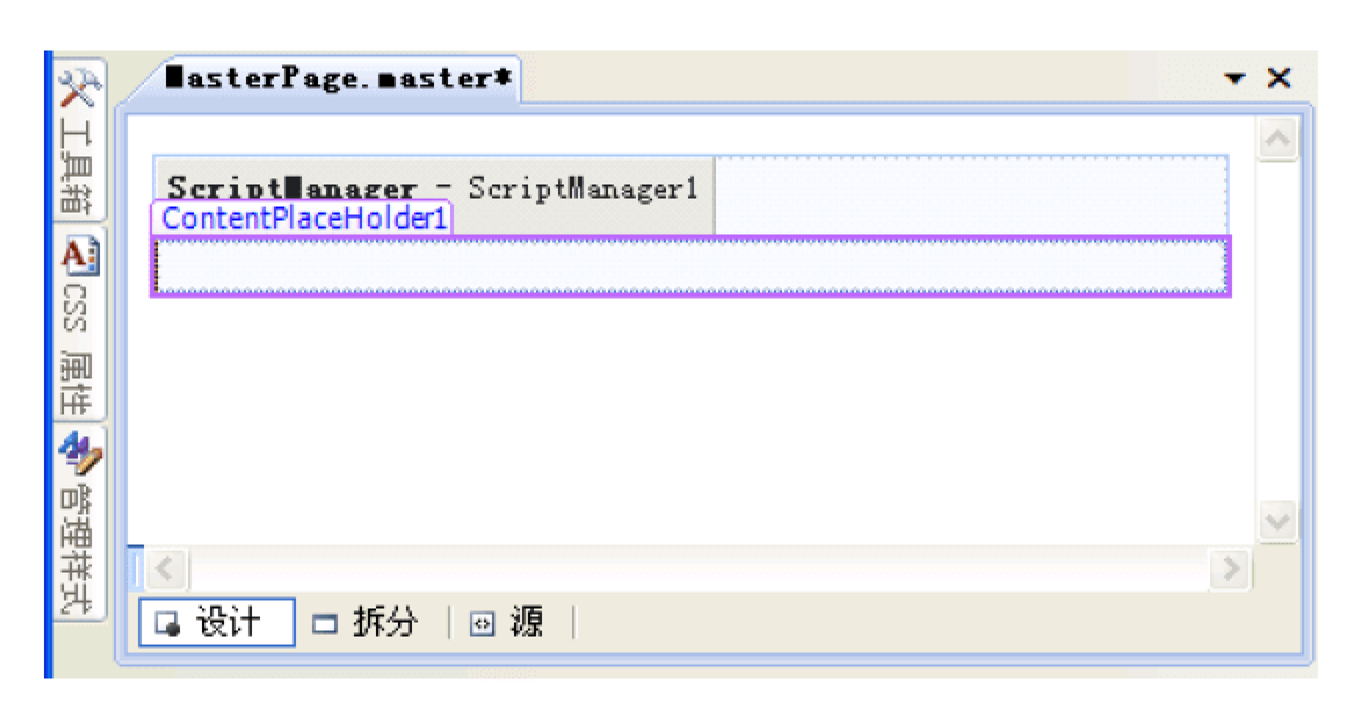


图 9-6　MasterPage.master 母版页设计视图

(4) 以 MasterPage.master 为母版，创建一个名为 ContentPage.aspx 的内容页。

(5) 在 ContentPage.aspx 页面中添加 1 个 UpdatePanel 控件，并在 UpdatePanel 控件中放置 1 个 Calendar 控件。如图 9-7 所示。

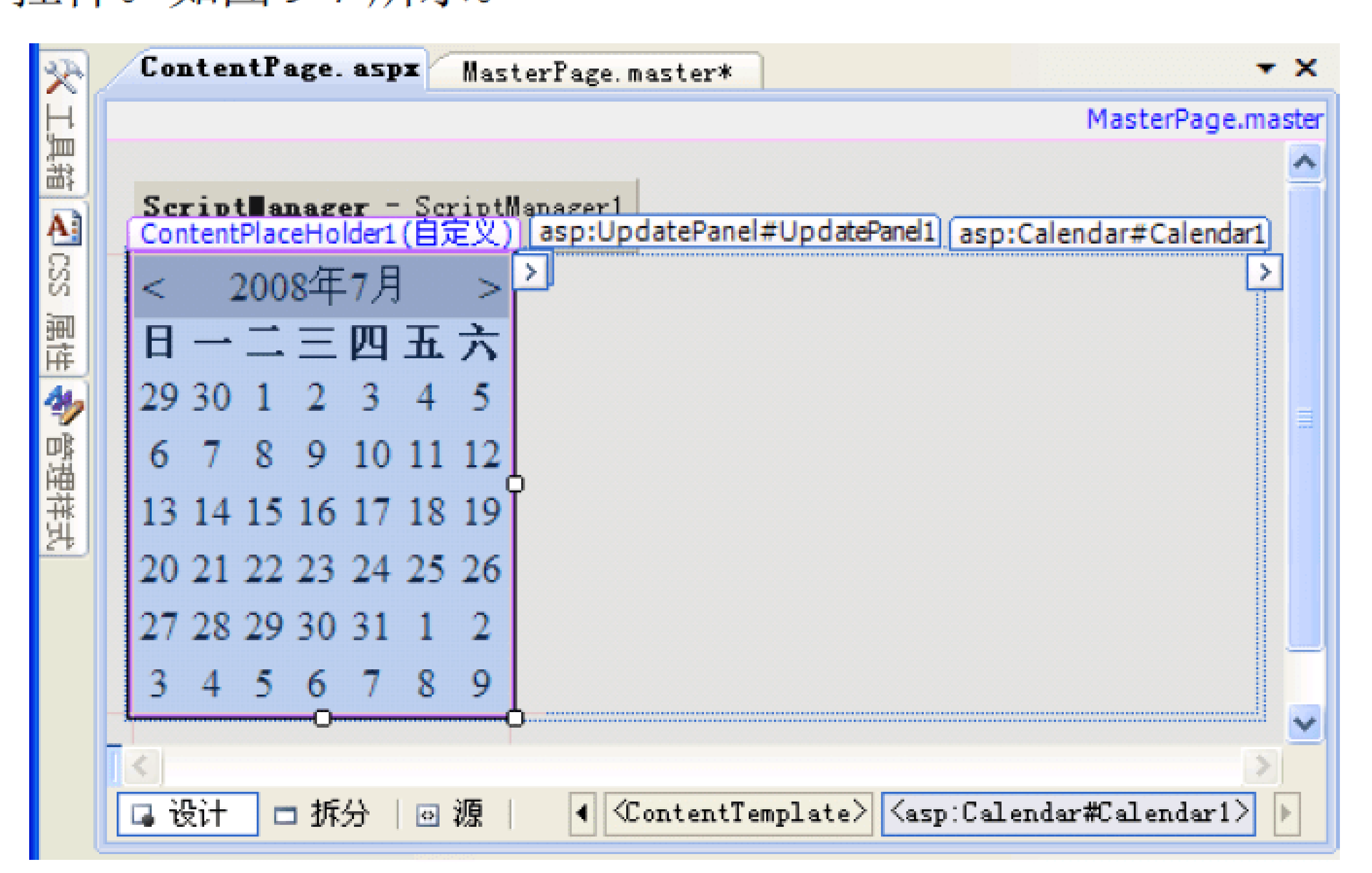


图 9-7　ContentPage. aspx 内容页设计视图

(6) 按 Ctrl+F5 组合键运行，当单击 Calendar 控件中的链接时，页面进行局部刷新。

2. 从母版页更新内容页的 UpdatePanel

下面具体通过示例介绍如何从母版页更新内容页的 UpdatePanel。

【例 9-6】在母版页中更新内容页的 UpdatePanel 的示例。

(1) 创建名称为 UpdatePanelExample5 的网站。

(2) 在【解决方案资源管理器】中，添加一个名为 MasterPage.master 的母版页。

(3) 将母版页 HTML 代码改写为如下内容：

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <asp:Button ID="IncrementButton" runat="server" Text="后一天"
            onclick="IncrementButton_Click" />
        <asp:Button ID="DecrementButton" runat="server" Text="前一天"
```

```
            onclick="DecrementButton_Click" />
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
```

(4) 以 MasterPage.master 为母版，创建一个名为 ContentPage.aspx 的内容页。

(5) 将内容页 HTML 代码改写为如下内容：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="IncrementButton" />
        <asp:AsyncPostBackTrigger ControlID="DecrementButton" />
    </Triggers>
</asp:UpdatePanel>
</asp:Content>
```

(6) 将 MasterPage.master 后台代码改写为如下内容：

```
protected void IncrementButton_Click(object sender, EventArgs e)
{
    //找到内容页中的 Calendar1 对象
    Calendar cal = ((Calendar)ContentPlaceHolder1.FindControl("Calendar1"));
    //将 Calendar1 对象日期增加一天
    cal.SelectedDate = cal.SelectedDate.AddDays(1) ;
}
protected void DecrementButton_Click(object sender, EventArgs e)
{
    //找到内容页中的 Calendar1 对象
    Calendar cal = ((Calendar)ContentPlaceHolder1.FindControl("Calendar1"));
    //将 Calendar1 对象日期减少一天
    cal.SelectedDate = cal.SelectedDate.AddDays(-1) ;
}
```

(7) 将 ContentPage.aspx 后台代码改写为如下内容：

```
protected void Page_Load(object sender, EventArgs e)
{
   //首次加载初始化 Calendar1 对象
```

```
        if (!IsPostBack)
            Calendar1.SelectedDate = DateTime.Today;
    }
```

(8) 按 Ctrl+F5 键运行，此时，单击母版页中的【前一天】和【后一天】按钮，触发了内容页中 UpdatePanel 控件的局部刷新，程序运行效果如图 9-8 所示。

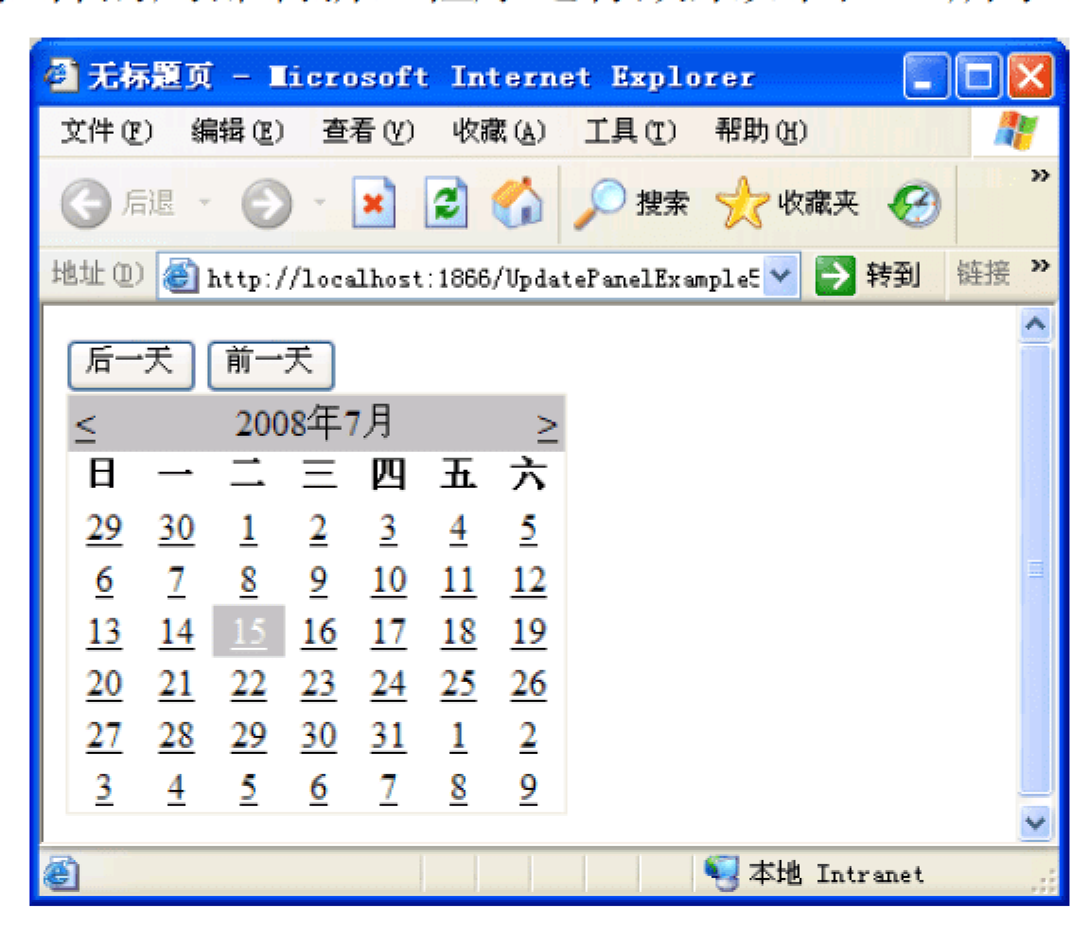


图 9-8　ContentPage 运行效果

9.3.6　UpdatePanel 异步更新中的错误处理

当发生 UpdatePanel 控件异步更新错误时，默认情况下会弹出一个错误对话框。如果设计者觉得不符合用户习惯，可以通过 ScriptManager 控件的 OnAsyncPostBackError 事件和 AsyncPostBackErrorMessage 属性捕捉和设置回传时的错误消息。

【例 9-7】将异步更新中的错误消息发送给客户端。

(1) 创建名称为 UpdatePanelExample6 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager 和一个 UpdatePanel 控件。

(3) 该页面设计如图 9-9 所示。

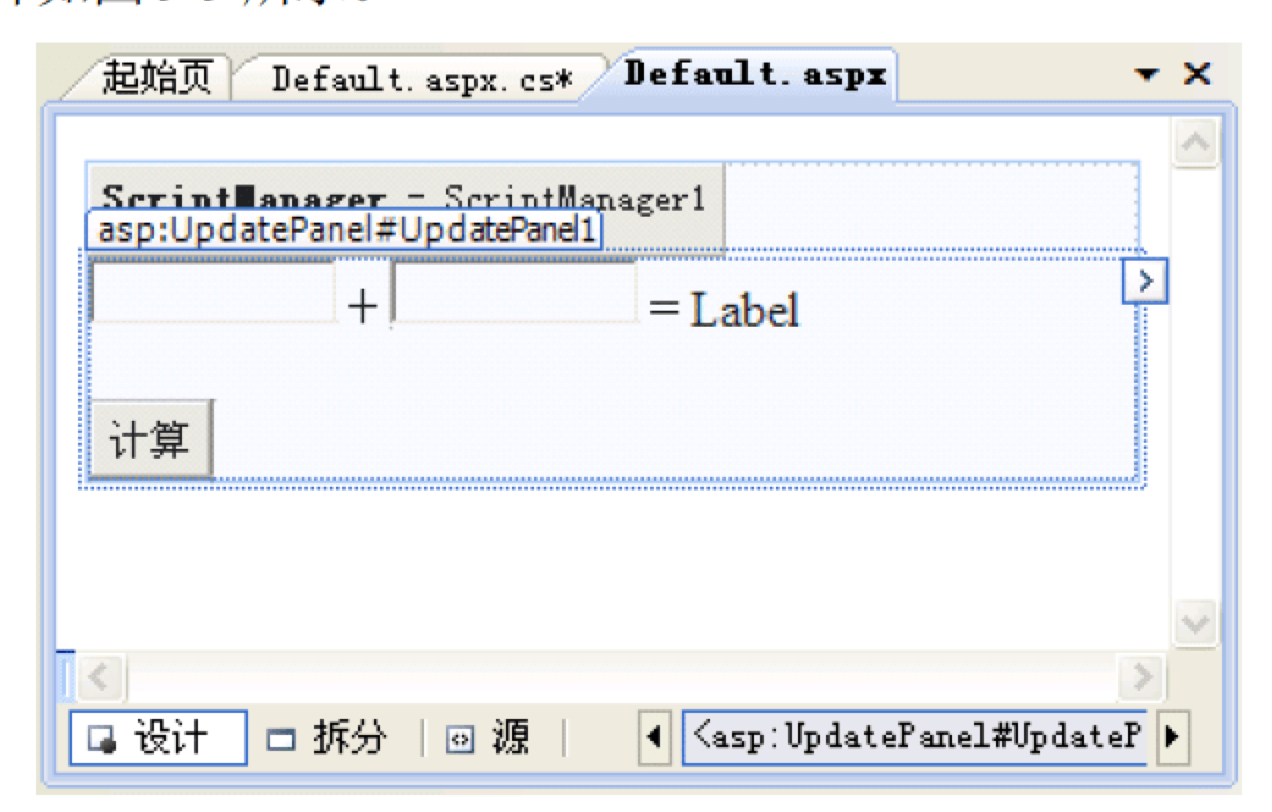


图 9-9　Default.aspx 的设计页面

对应【源】视图中的代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
                onasyncpostbackerror="ScriptManager1_AsyncPostBackError">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:TextBox ID="TextBox1" runat="server" Width="80px"></asp:TextBox>
     <asp:Label ID="Label1" runat="server" Text="+" ></asp:Label>
     <asp:TextBox ID="TextBox2" runat="server" Width="80px"></asp:TextBox>
     <asp:Label ID="Label2" runat="server" Text="="></asp:Label>
     <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label> <br /> <br />
     <asp:Button ID="Button1" runat="server" Text="计算" onclick="Button1_Click" />
    </ContentTemplate>
 </asp:UpdatePanel>
```

(4) 添加后台代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        Double a = Convert.ToDouble(TextBox1.Text);
        Double b = Convert.ToDouble(TextBox2.Text);
        Double res = a * b;
        Label3.Text = res.ToString("f2");
    }
    catch (Exception ex)
    {
        if (TextBox1.Text.Length >= 0 && TextBox2.Text.Length >= 0)
        {
            ex.Data["ExtraInfo"] = "这两个数无法相乘";
        }
        throw ex;
    }
}
protected void ScriptManager1_AsyncPostBackError(object sender, AsyncPostBackErrorEventArgs e)
{
    if (e.Exception.Data["ExtraInfo"] != null)
    {
        ScriptManager1.AsyncPostBackErrorMessage = e.Exception.Data["ExtraInfo"].ToString();
    }
    else
    {
        ScriptManager1.AsyncPostBackErrorMessage = e.Exception.Message;
    }
```

```
}
```

(5) 按 Ctrl+F5 键运行，当相乘的两个数中有非法数字时，弹出自定义错误信息的对话框。

如果按 F5 键调试运行，调试器会首先捕捉到异常，从而中断代码的执行，所以要求程序在不调试的情况下运行。

9.4　UpdateProgress 控件

UpdateProgress 控件一般与 UpdatePanel 控件联合使用，即在 UpdatePanel 异步更新过程中，显示提示信息。这些信息可以是一段文字、进度条或各种动画。当异步更新完成时，提示信息自动消失。

9.4.1　UpdateProgress 控件属性

UpdateProgress 控件常用属性如表 9-3 所示。

表 9-3　UpdateProgress 控件常用属性

属　　性	说　　明
AssociateUpdatePanelID	设置哪个 UpdatePanel 控件产生的回送会显示 UpdateProgress 的内容。
DisplayAfter	当引发回送后多少毫秒会显示 UpdateProgress 控件的内容。
DynamicLayout	设置 UpdateProgress 控件的显示方式。true--当 UpdateProgress 控件不显示的时候不占用空间(默认)。false--当 UpdateProgress 控件不显示的时候仍然占用空间。

如果没有设定 UpdateProgress 控件的 AssociateUpdatePanelID 属性，则任何一个异步更新都会使 UpdateProgress 控件显示出来。相反，如果将 UpdateProgress 控件的 AssociateUpdatePanelID 属性设为某个 UpdatePanel 控件的 ID，那只有该 UpdatePanel 控件引发的异步更新才会使相关联的 UpdateProgress 控件显示出来。

9.4.2　使用 UpdateProgress 控件的方法

在下面的【例 9-8】中页面上有一个 UpdatePanel 控件和一个 UpdateProgress 控件，当 UpdatePanel 控件异步更新时，显示 UpdateProgress 控件的提示内容。

【例 9-8】使用 UpdateProgress 控件示例。

(1) 创建名称为 UpdateProgressExample1 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、一个 UpdatePanel 和一个 UpdateProgress 控件。

(3) 在 UpdatePanel 中，添加一个 Label 控件和一个 Button 控件。
(4) 在 UpdateProgress 中，添加“正在更新…”文字和进度条图像。
(5) 该页面设计如图 9-10 所示。

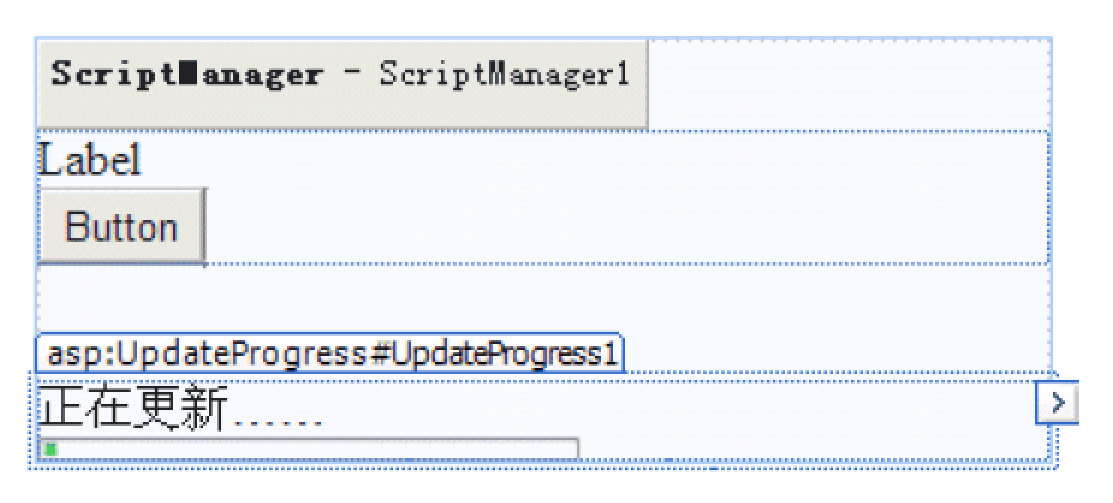


图 9-10　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>    <br />
        <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
        <br />
    </ContentTemplate>
</asp:UpdatePanel>    <br /> <br />
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
    <ProgressTemplate>
            正在更新……<br />
            <asp:Image ID="Image1" runat="server" ImageUrl="strip.gif" Width="180px" /><br />
    </ProgressTemplate>
</asp:UpdateProgress>
```

(6) 添加后台代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    //延迟 3 秒
    System.Threading.Thread.Sleep(3000);
    Label1.Text = DateTime.Now.ToString();
}
```

(7) 按 Ctrl+F5 键运行，观察程序效果。

9.4.3　使用多个 UpdateProgress 控件的方法

下面的【例 9-9】，演示了页面上使用多个 UpdateProgress 控件的方法，该示例的页面上放置两个 UpdatePanel 和两个 UpdateProgress 控件，将 UpdateProgress1 控件的 AssociateUpdatePanelID 属性设为 UpdatePanel1 控件的 ID，UpdateProgress2 控件的 AssociateUpdatePanelID 属性设为 UpdatePanel2 控件的 ID，当 UpdatePanel1 异步更新时，

UpdateProgress1 显示，而当 UpdatePanel2 异步更新时，显示 UpdateProgress2 的提示信息。

【例 9-9】使用多个 UpdateProgress 控件。

(1) 创建名称为 UpdateProgressExample2 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、两个 UpdatePanel 和两个 UpdateProgress 控件。

(3) 在 UpdatePanel 控件和 UpdateProgress 控件中，添加与【例 9-8】一样的控件。

(4) 该页面设计如图 9-11 所示。

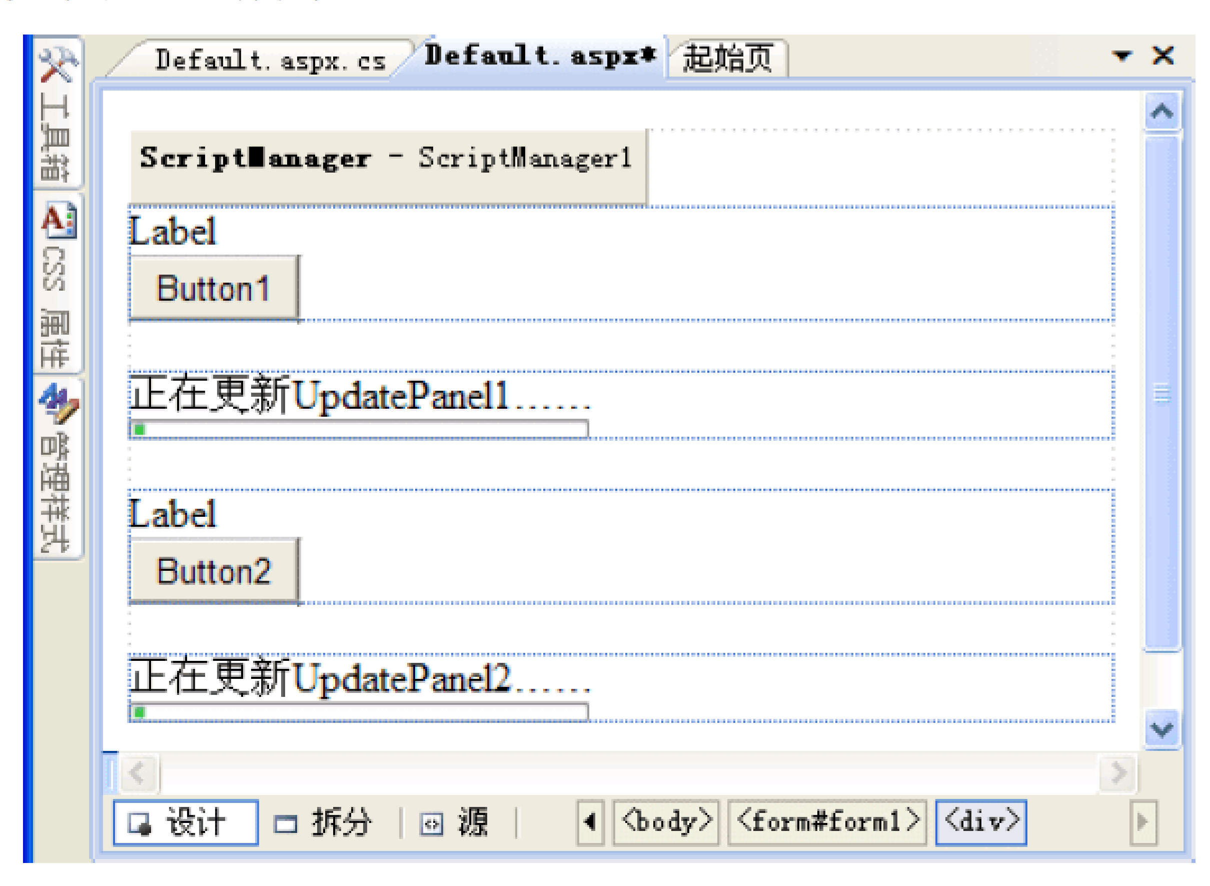


图 9-11　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>      <br />
        <asp:Button ID="Button1" runat="server" Text="Button1" onclick="Button1_Click" /><br />
    </ContentTemplate>
</asp:UpdatePanel>      <br />
<asp:UpdateProgress ID="UpdateProgress1" runat="server"
AssociatedUpdatePanelID="UpdatePanel1">
      <ProgressTemplate>
            正在更新 UpdatePanel1……<br />
            <asp:Image ID="Image1" runat="server" ImageUrl="strip.gif" Width="180px" /><br />
      </ProgressTemplate>
</asp:UpdateProgress>        <br />
<asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode="Conditional">
        <ContentTemplate>
            <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>        <br />
            <asp:Button ID="Button2" runat="server" Text="Button2" onclick="Button2_Click" /> <br
/>
        </ContentTemplate>
</asp:UpdatePanel>      <br />
```

```
        <asp:UpdateProgress ID="UpdateProgress2" runat="server"
AssociatedUpdatePanelID="UpdatePanel2">
            <ProgressTemplate>
                正在更新 UpdatePanel2……<br />
                <asp:Image ID="Image2" runat="server" ImageUrl="strip.gif" Width="180px" /><br />
            </ProgressTemplate>
        </asp:UpdateProgress>
```

(5) 添加后台代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);
    Label1.Text = DateTime.Now.ToString();
}
protected void Button2_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);
    Label2.Text = DateTime.Now.ToString();
}
```

(6) 按 Ctrl＋F5 键运行，观察异步更新时 UpdateProgress 控件的显示效果。

虽然一个页面允许有多个 UpdateProgress 控件，但是在实际中，一般在一个页面中应该只放置一个 UpdateProgress。

9.5　Timer 控件

Timer 控件是 ASP.NET AJAX 中又一个重要的服务器控件。通过它可以完成局部页面的定时更新，从而实现局部页面定时刷新、图片自动播放、超时自动退出等功能。

9.5.1　Timer 控件属性和事件

Timer 控件常用属性和事件如表 9-4 所示。

表 9-4　Timer 控件常用属性和事件

属性和事件	说　明
Interval 属性	用于指定间隔时间
Tick 事件	指定间隔到期后触发
Enabled 属性	用于表示是否允许 tick 事件

Interval 属性用来决定每隔多长时间要引发回传，其设置值的单位是毫秒，默认值则是 60000 毫秒，也就是 60 秒。需要注意的是，将 Timer 控件的 Interval 属性设置成较小的值会

使得回送频率增加，也很容易使得 Web 服务器的流量大增，对整体资源耗用与效率都会造成不良的影响。因此尽量在确实需要的时候使用 Timer 控件来定时更新页面上的内容。

每当 Timer 控件的 Interval 属性所设置的间隔时间到达而进行回传时，就会在服务器上引发 Tick 事件，执行定时操作。

另外一个常用的属性就是 Enabled，用户可以将 Enabled 属性设置成 false 以便让 Timer 控件停止计时，而当需要让 Timer 控件再次开始计时的时候，只需再将 Enabled 属性设置成 True 即可。

Timer 控件在 UpdatePanel 控件的内外是有区别的。当 Timer 控件在 UpdatePanel 控件内部时，JavaScript 计时组件只有在一次回传完成后才会重新建立。也就是说直到网页回传完成之前，定时器间隔时间不会从头计算。例如，用户设置 Timer 控件的 Interval 属性值为 3000ms(3 秒)，但是回传操作本身却花了 2 秒才完成，则下一次的回传将发生在前一次回传被引发之后的 5 秒。而如果 Timer 控件位于 UpdatePanel 控件之外，则当回传正在处理时，下一次回传将发在前一次回传被引发之后的 3 秒。也就是说，UpdatePanel 控件的内容被更新之后的 1 秒，就会再次看到该控件被更新。

9.5.2　使用 Timer 控件定时更新 UpdatePanel

下面介绍一个在 UpdatePanel 内部使用 Timer 控件的简单示例，该示例实现图片的自动刷新。

【例 9-10】在 UpdatePanel 内部使用 Timer 控件。

(1) 创建名称为 TimerExample1 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、一个 UpdatePanel。

(3) 在 UpdatePanel 控件中，添加一个 Timer 控件和一个 Image 控件。

(4) 该页面设计如图 9-12 所示。

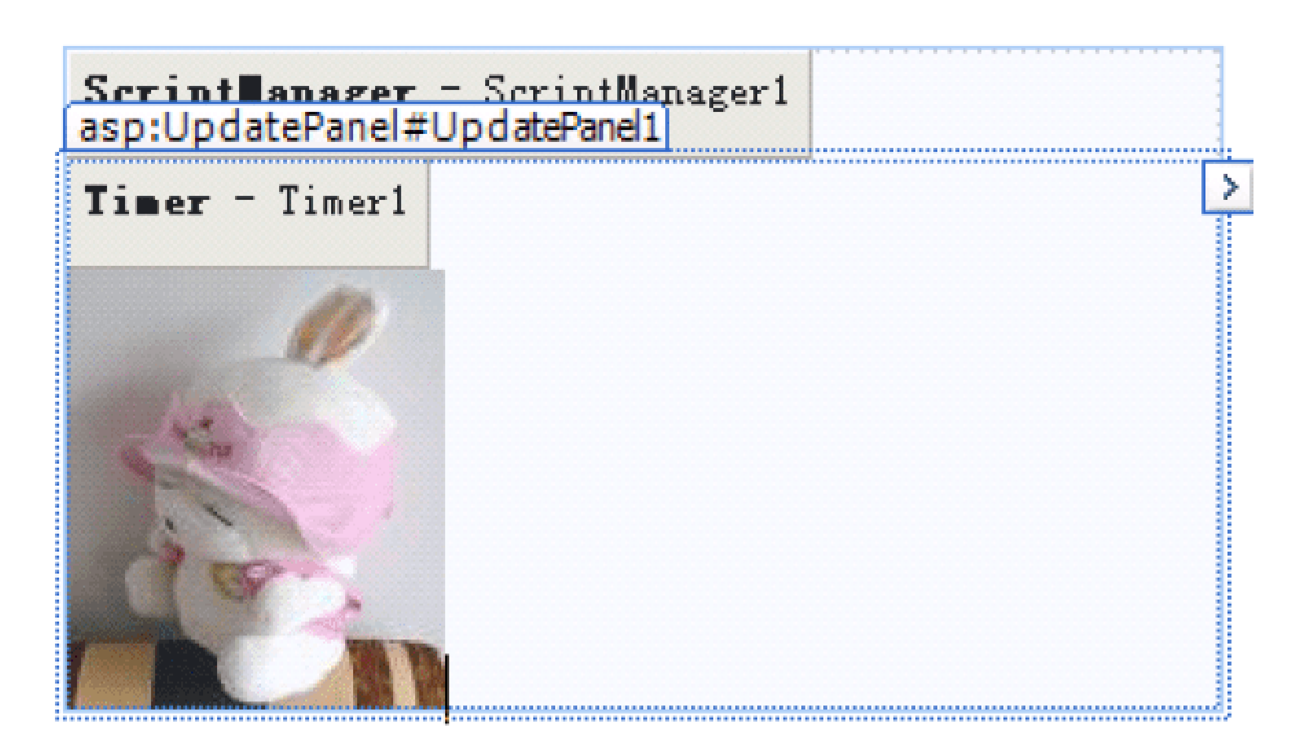


图 9-12　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Timer ID="Timer1" runat="server" Interval="1000" ontick="Timer1_Tick">
        </asp:Timer>
        <asp:Image ID="Image1" runat="server" ImageUrl="~/image/1.jpg"/>
    </ContentTemplate>
</asp:UpdatePanel>
```

(5) 添加后台代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Page.IsPostBack == false)
   {
       //设置网页上的变量
       ViewState["number"] = 1;
   }
}
protected void Timer1_Tick(object sender, EventArgs e)
{
   ViewState["number"] = (int)ViewState["number"] % 9 + 1;
   Image1.ImageUrl = string.Format("~/image/{0}.jpg", ViewState["number"]);
}
```

(6) 按 Ctrl+F5 键运行，可以看到每间隔 1 秒局部刷新一幅图片。

9.5.3　使用 Timer 控件定时更新多个 UpdatePanel

在下面的【例 9-11】中，Timer 控件定时更新 2 个 UpdatePanel 控件，Timer 控件被放在 UpdatePanel 控件的外面，并将它配置为 UpdatePanel 的触发器。

【例 9-11】使用 Timer 控件定时更新多个 UpdatePanel 示例。

(1) 创建名称为 TimerExample2 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、两个 UpdatePanel。

(3) 在所有 UpdatePanel 控件之外，添加一个 Timer 控件，并设置它为 UpdatePanel 的异步触发器。

(4) 该页面设计如图 9-13 所示。

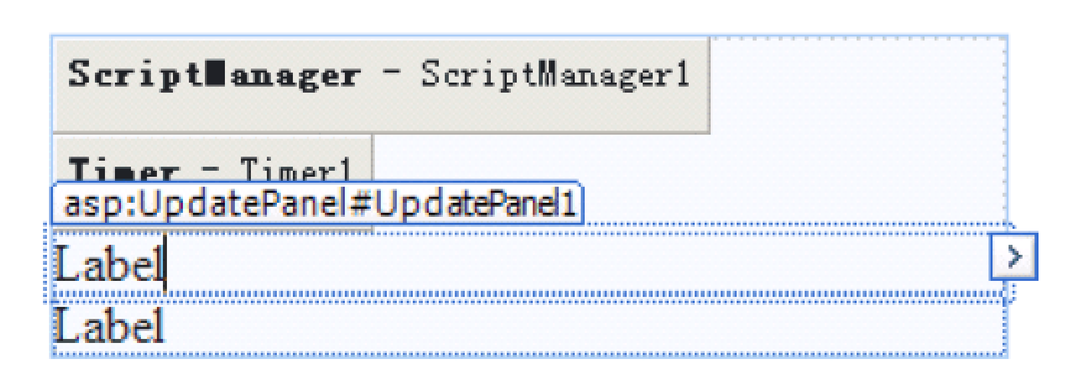


图 9-13　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:Timer ID="Timer1" runat="server" Interval="2000" ontick="Timer1_Tick">
</asp:Timer>
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="Timer1" />
    </Triggers>
</asp:UpdatePanel>
<asp:UpdatePanel ID="UpdatePanel2" runat="server" UpdateMode="Conditional">
    <ContentTemplate>
        <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="Timer1" />
    </Triggers>
</asp:UpdatePanel>
```

(5) 添加后台代码如下：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel1 更新于： " + DateTime.Now.ToLongTimeString();
    Label2.Text = "UpdatePanel2 更新于： " + DateTime.Now.ToLongTimeString();
}
```

(6) 按 Ctrl＋F5 键运行，可以看到每间隔 2 秒同时刷新两个 UpdatePanel。

9.6　ASP.NET AJAX Control Extenders 扩展控件

ASP.NET AJAX Control Extenders 是一些派生自 System.Web.UI.ExtenderControl 基类的控件，通常后缀为 Extender，扩展控件必须和被其控制的控件组合才能发挥作用。

利用扩展控件可以为页面中已存在的控件添加其他功能(一般都是 AJAX 或者 JavaScript 支持)。它们使得开发者可以优美地封装控件行为，并且它们使得为应用程序添加更丰富的功能变得非常简单。

如图 9-14 所示，利用 CalendarExtender 日期扩展控件，使得 TextBox 控件录入日期更加方便、直观，而且增加了 AJAX 功能。

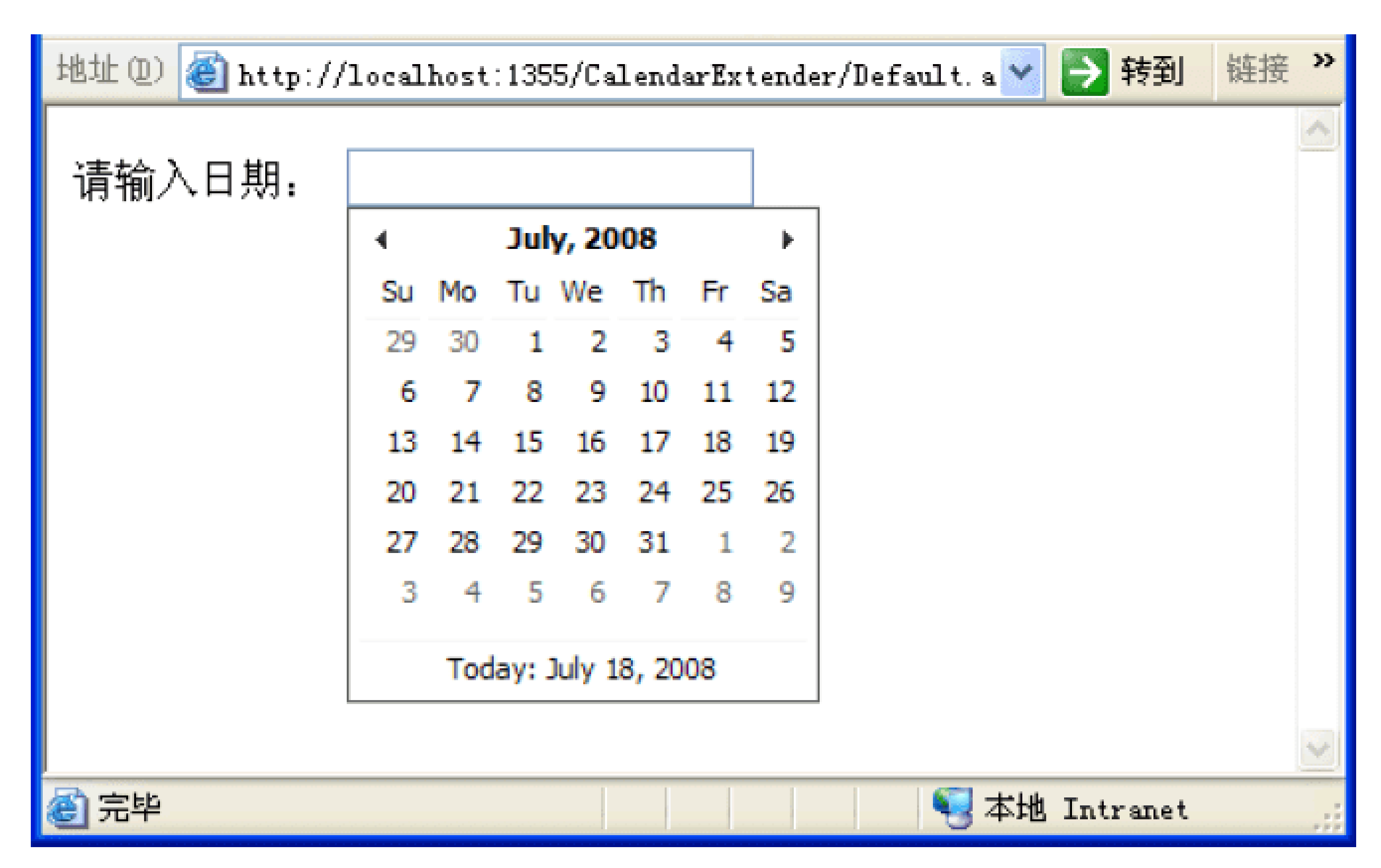


图 9-14　TextBox 和 CalendarExtender 组合

TextBox 和 CalendarExtender 组合的代码如下：

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<cc1:CalendarExtender ID="TextBox1_CalendarExtender" runat="server"
        Enabled="True" TargetControlID="TextBox1">
 </cc1:CalendarExtender>
```

其中将 CalendarExtender 控件的 TargetControlID 属性指向 TextBox 控件的 ID。

9.6.1　如何使用 ASP.NET AJAX Control Extenders

ASP.NET AJAX Control Toolkit 是一个非常好的扩展控件工具包，是由 Microsoft 公司和其他开发人员共同开发的，它包括四十多个免费的扩展控件，而且一直都在增加，可以轻松地被下载并添加到 VWD 2008 的工具箱。

1. 添加扩展控件到 VWD 2008 的工具箱

在 VWD 2008 工具箱中，添加扩展控件的步骤如下：

(1) 从 Microsoft 公司网站上，下载 ASP.NET AJAX Control Toolkit For .NET3.5 工具包。

(2) 将该工具包解压缩。

(3) 在 VWD 2008 工具箱中，新建一个名为 AJAX Control Toolkit 的选项卡。

(4) 在解压后的工具包中找到 AJAXControlToolkit.dll 文件，将其拖放到 VWD 2008 工具箱的 AJAX Control Toolkit 选项卡中。

这样，扩展控件就会出现在 VWD 2008 的工具箱中，如图 9-15 所示。

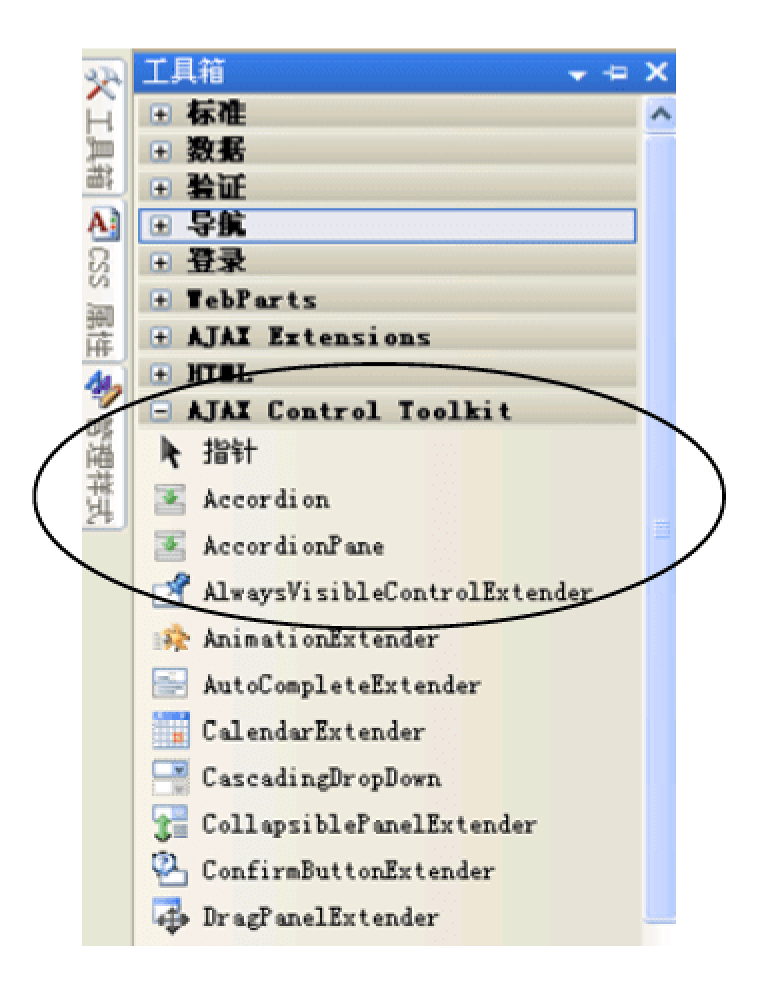


图 9-15　VWD 2008 的工具箱

2. 绑定扩展控件到某个已存在的控件

扩展控件被添加到 VWD 2008 工具箱后，再选择控件时，会发现一个新的【添加扩展程序】任务选项出现在被选择的控件上，如图 9-16 所示。

如果单击【添加扩展程序】任务选项，将会弹出如图 9-17 所示的对话框，它包含了所有可以选择的扩展控件，确定后就完成了扩展控件的绑定。

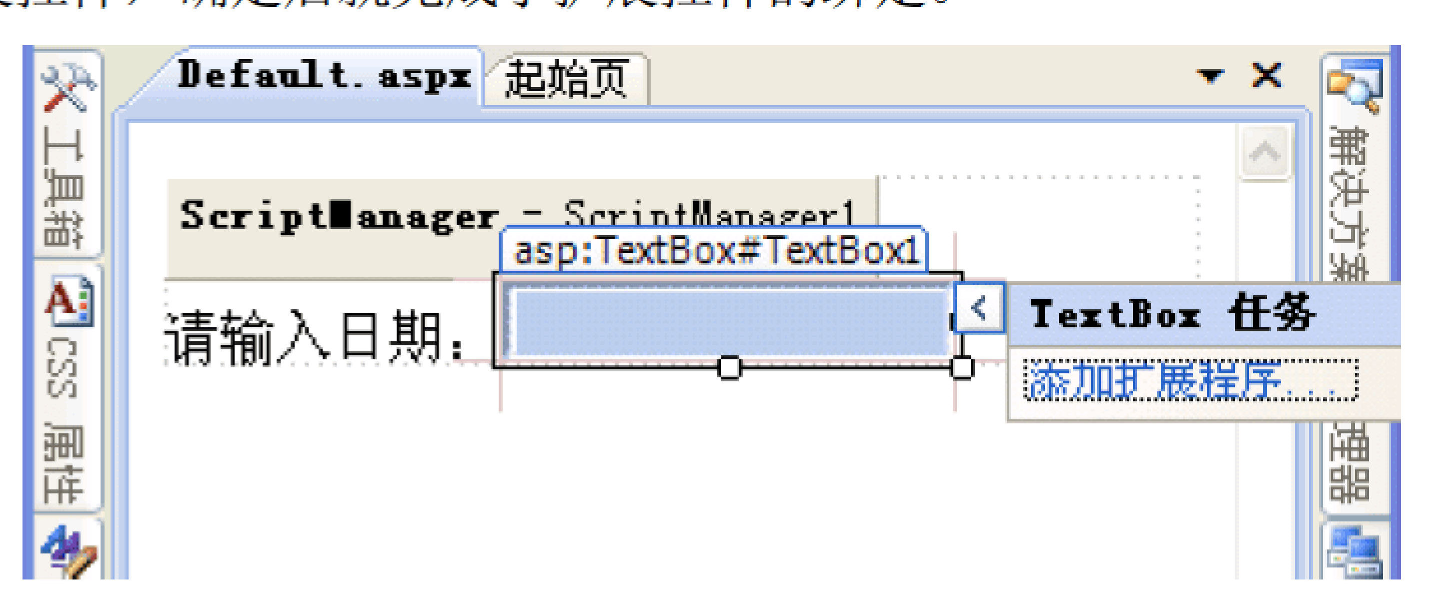


图 9-16　添加扩展控件

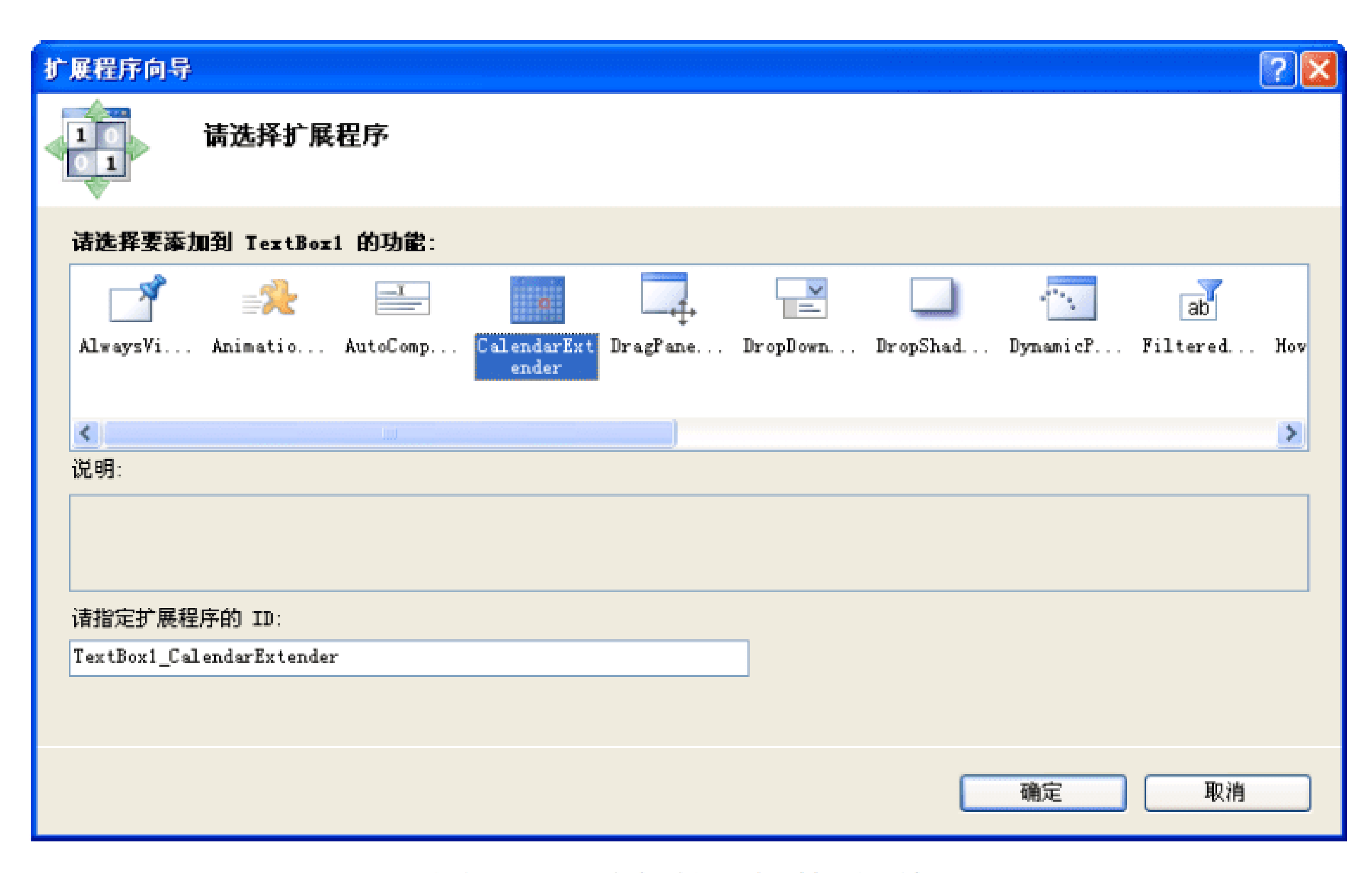


图 9-17　选择扩展控件对话框

3. 合并属性窗格

在 VWD 2008 中，当将一个扩展控件绑定到一个控件后，该控件的属性窗格也会扩展显示扩展控件的属性。如图 9-18 所示。

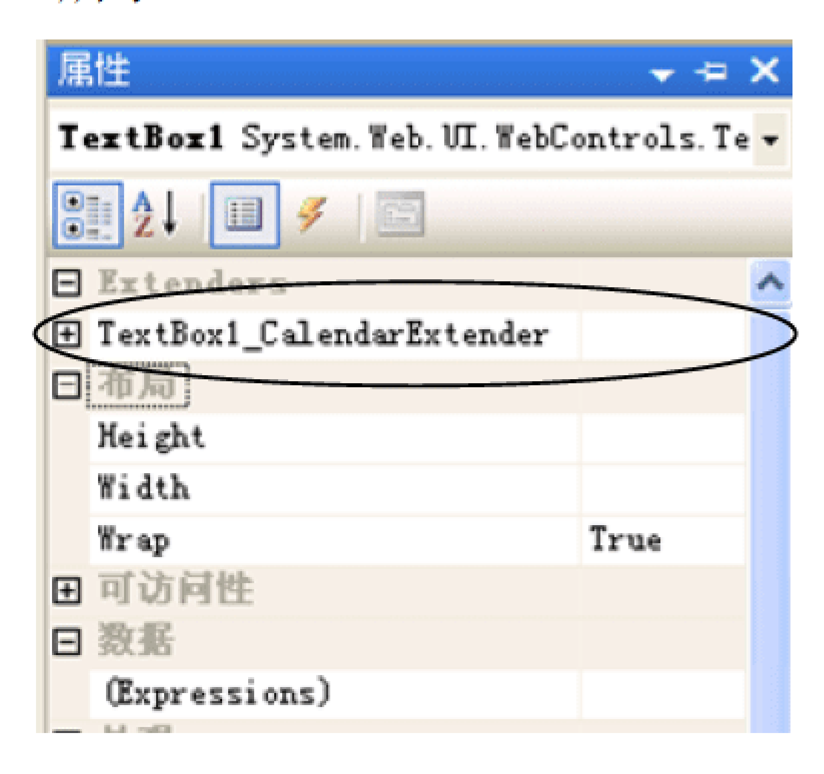


图 9-18　扩展显示属性窗格

4. 删除绑定的扩展控件

如果想删除某个控件的扩展控件，只需要从控件的任务选项中选择“删除扩展程序”选项，如图 9-19 所示。

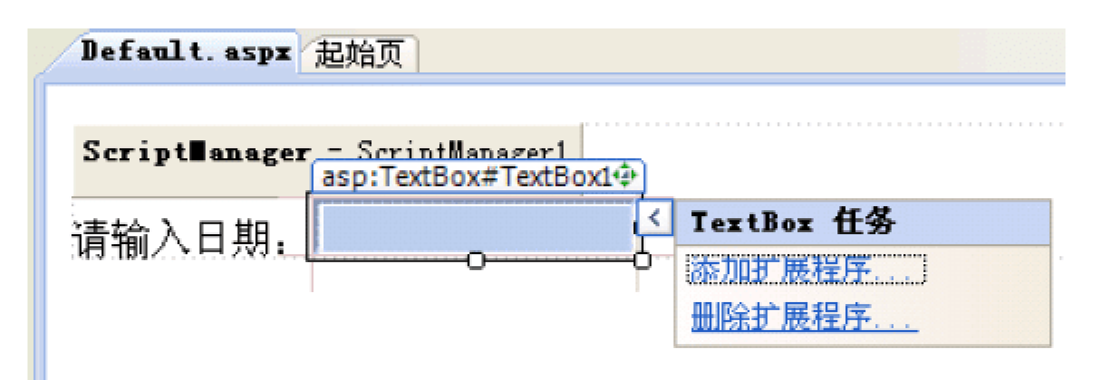


图 9-19　删除扩展控件

9.6.2　使用 FilteredTextBoxExtender 控件防止输入非法字符

FilteredTextBoxExtender 扩展控件是用来对文本框进行过滤的，让文本框只能对设定的值进行输入。输入类型有 Numbers(数字)，LowercaseLetters(小写字母)，UppercaseLetters (大写字母)和 Custom(自定义字符串)。FilteredTextBoxExtender 扩展控件的常用属性如表 9-5 所示。

表 9-5　FilteredTextBoxExtender 常用属性

属　　性	说　　明
TargetControlID	要进行过滤的目标 TextBox 的 ID
FilterType	字符过滤类型，提供的有如下 4 种：Numbers，LowercaseLetters，UppercaseLetters，Custom。它们之间可以同时指定多个类型，如：FilterType="Custom, Numbers"
ValidChars	当 FilterType 为 Custom 时允许输入的字符，否则将被忽略，如：ValidChars="+-=/*()."

下面通过具体示例演示 FilteredTextBoxExtender 扩展控件的用法。

【例 9-12】演示 FilteredTextBoxExtender 扩展控件的用法。

(1) 创建名称为 FilteredTextBoxExtender 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、一个 TextBox 和一个 Label。

(3) 单击 TextBox 控件的【添加扩展程序】任务选项，绑定 FilteredTextBoxExtender 扩展控件。

(4) 该页面设计如图 9-20 所示。



图 9-20　Default.aspx 的设计页面

对应【源】视图中的部分代码如下：

```
<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit" tagprefix="cc1" %>
    •••
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:Label ID="Label1" runat="server" Text="只能输入大写字母和数字："></asp:Label>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <cc1:FilteredTextBoxExtender ID="TextBox1_FilteredTextBoxExtender"
runat="server" Enabled="True" FilterType="Numbers,UppercaseLetters" TargetControlID="TextBox1">
    </cc1:FilteredTextBoxExtender>
```

(5) 按 Ctrl+F5 键运行，可以看到 TextBox 控件只允许输入大写字母和数字。

由于使用文本框过滤是在客户端进行的，所以使用 FilteredTextBoxExtender 扩展控件的网页中必须包含 ScriptManager 控件。

9.6.3　使用 SlideShowExtender 控件播放幻灯片

SlideShowExtender 扩展控件用于控制 Image 控件，它可以很方便地实现自动和手动播放图片功能。由于播放图片是在客户端进行的，所以使用 SlideShowExtender 扩展控件的网页中必须包含 ScriptManager 控件。

SlideShowExtender 扩展控件的常用属性如表 9-6 所示。

表 9-6　SlideShowExtender 常用属性

属　　性	说　　明
TargetControlID	该控件的目标扩展控件
ImageDescriptionLabelID	对显示的图片进行说明的 Label 控件
Loop	是否为图片进行循环放映
NextButtonID	控制显示下一张图片的按钮
PlayButtonID	控制进行播放或停止的按钮
PlayButtonText	当 Image 中的图片在放映时，PlayButtonID 按钮显示的文本
PreviousButtonID	控制显示前一张图片的按钮
StopButtonText	当 Image 中的图片停止放映时，PlayButtonID 按钮显示的文本
PlayInterval	播放每幅图片的间隔，单位毫秒，默认 3000 毫秒(3 秒)
SlideShowServiceMethod	进行幻灯片式放映时加载图片的方法

下面通过具体示例演示 SlideShowExtender 扩展控件的用法。

【例 9-13】演示 SlideShowExtender 扩展控件的用法。

(1) 创建名称为 SlideShowExtender 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager、一个 Image、一个 Label 和 3 个 Button 控件。Image 控件用于显示图片，Label 控件用于显示图片说明，Button 控件用于控制图片显示。

(3) 单击 Image 控件的【添加扩展程序】任务选项，绑定 SlideShowExtender 扩展控件。

(4) 该页面【源】视图中的部分代码如下：

```
<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit" tagprefix="cc1" %>
        •••
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:Image ID="Image1" runat="server"
                    Height="300"
                    Style="border: 1px solid black;width:auto"
                    ImageUrl="~/images/Blue hills.jpg"
                    AlternateText="Blue Hills image" />
        <asp:Label runat="server" ID="imageDescription" ></asp:Label><br /><br />
        <asp:Button runat="Server" ID="prevButton" Text="前一个"   />
        <asp:Button runat="Server" ID="playButton" Text="开始"   />
        <asp:Button runat="Server" ID="nextButton" Text="后一个"   />

        <cc1:SlideShowExtender   ID="slideshowextend1" runat="server"
                    TargetControlID="Image1"
                    SlideShowServiceMethod="GetSlides"
                    AutoPlay="true"
                    ImageDescriptionLabelID="imageDescription"
```

```
            NextButtonID="nextButton"
            PlayButtonText="开始"
            StopButtonText="停止"
            PreviousButtonID="prevButton"
            PlayButtonID="playButton"
            Loop="true" PlayInterval="2000">
</cc1:SlideShowExtender>
```

(5) 添加后台代码如下：

```
[System.Web.Services.WebMethod]
[System.Web.Script.Services.ScriptMethod]
public static AjaxControlToolkit.Slide[] GetSlides()
{
    return new AjaxControlToolkit.Slide[] {
        new AjaxControlToolkit.Slide("images/Blue hills.jpg", "", "Go Blue"),
        new AjaxControlToolkit.Slide("images/Sunset.jpg", "", "Setting sun"),
        new AjaxControlToolkit.Slide("images/Winter.jpg", "", "Wintery..."),
        new AjaxControlToolkit.Slide("images/Water lilies.jpg", "", "Lillies in the water"),
        new AjaxControlToolkit.Slide("images/VerticalPicture.jpg", "", "Portrait style icture")};
}
```

GetSlides 方法创建了一个 AjaxControlToolkit.Slide 类型的数组，该数组包含了所有要播放的图片，SlideShowExtender 控件调用 GetSlides 方法，得到这些图片，并将其在 Image 控件中依次显示出来。

(6) 按 Ctrl＋F5 键运行，运行效果如图 9-21 所示。

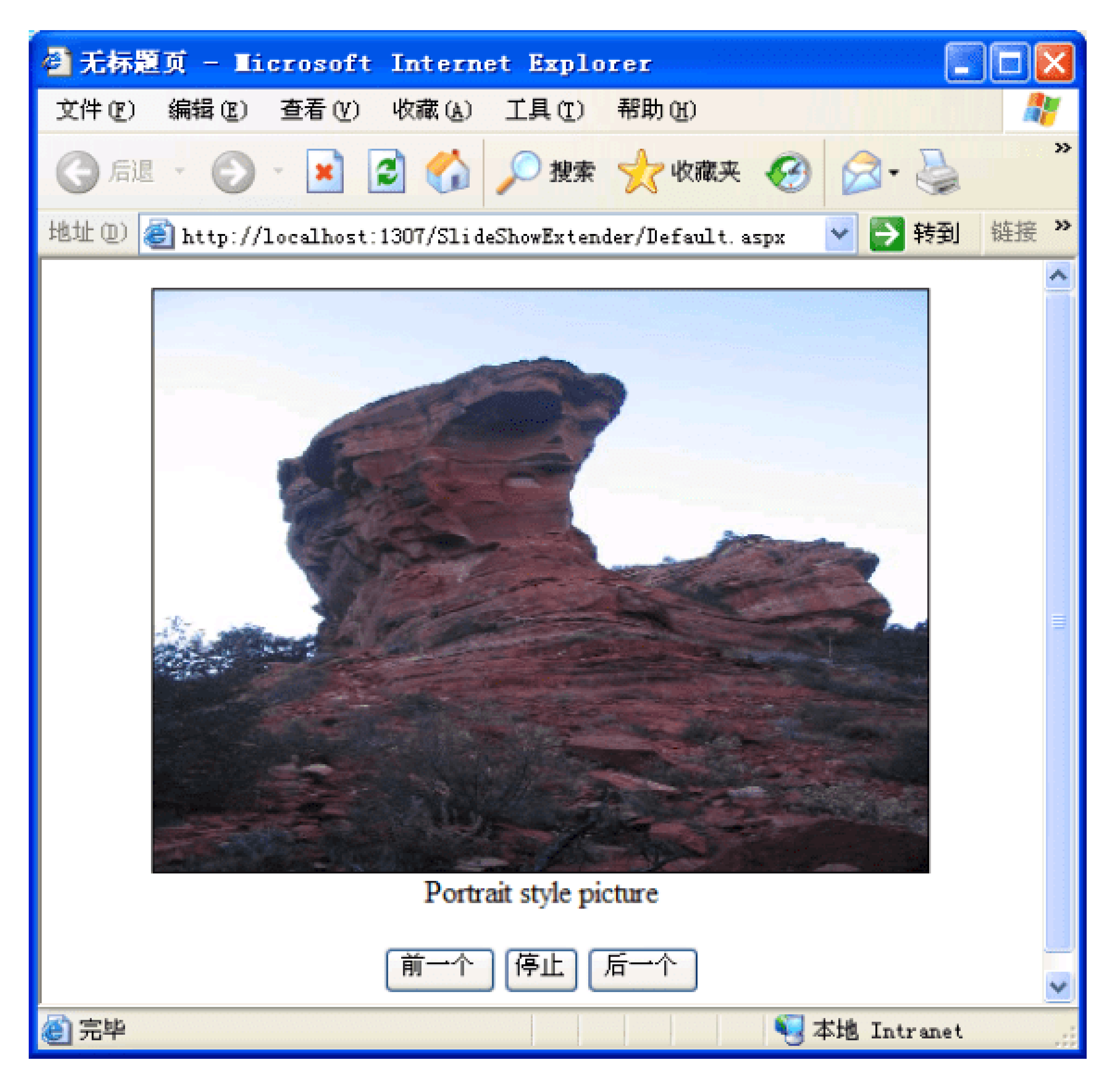


图 9-21　【例 9-13】的运行效果

9.6.4　使用 AlwaysVisibleControlExtender 固定位置显示控件

AlwaysVisibleControlExtender 扩展控件使其被控制的控件显示在页面的固定位置上，常用于漂浮在网页两边，跟随着滚动条滚动的悬浮广告等。AlwaysVisibleControlExtender 是一个非常简单的扩展控件，不用编码只需简单的设置就可使用。

AlwaysVisibleControlExtender 扩展控件的常用属性如表 9-7 所示。

表 9-7　AlwaysVisibleControlExtender 常用属性

属　性	说　明
TargetControlID	目标控件 ID，要浮动的控件
HorizontalOffset	距离浏览器的水平边距，默认值 0px
HorizontalSide	水平停靠方向，默认值 Left
VerticalOffset	距离浏览器的垂直边距，默认值 0px
VerticalSide	垂直停靠方向，默认值 Top
ScrollEffectDuration	滚动效果的延迟时间，单位为秒，默认值 0.1

下面通过具体示例演示 AlwaysVisibleControlExtender 扩展控件的用法。

【例 9-14】演示 AlwaysVisibleControlExtender 扩展控件的用法。

(1) 创建名称为 AlwaysVisibleControlExtender 的网站。

(2) 在 Default.aspx 的【设计】视图中，放置一个 ScriptManager 和一个 Panel。在 Panel 中添加一个 ImageButton 控件。

(3) 单击 Panel 控件的【添加扩展程序】任务选项，绑定 AlwaysVisibleControlExtender 扩展控件。

(4) 该页面【源】视图中的部分代码如下：

```
<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit" tagprefix="cc1" %>
    …
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:Panel ID="Panel1" runat="server"    Width="150px" height="120px" >
          <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~/1.jpg" />
    </asp:Panel>
    <cc1:AlwaysVisibleControlExtender ID="avce" runat="server"
                TargetControlID="Panel1"
                VerticalOffset="10"
                HorizontalOffset="10"
                ScrollEffectDuration=".1" />
```

(5) 按 Ctrl+F5 键运行，运行效果如图 9-22 所示，上下移动浏览器的滚动条，可以看到，图片始终保持在页面的左上角位置上。



图 9-22　程序运行效果

ASP.NET AJAX Control Toolkit 是一个免费开源的工具包，包含了新的 Web 服务器控件，随着控件数量的逐渐增加，无疑会给 Web 应用程序开发带来更大方便。

实验 9　ASP.NET AJAX

1. 实验目的

进一步熟悉 ASP.NET AJAX 技术，掌握 ASP.NET AJAX 服务器控件和扩展控件的使用方法。

2. 实验内容和要求

(1) 新建名字为 AJAX_Exercise 的网站。

(2) 在网站中建立名为 Images 的文件夹，并在文件夹中添加几张图片。

(3) 添加一个网页，当单击 Button 控件时，局部更新 Image 控件中的图片。

(4) 添加一个网页，当单击 Button 控件时，局部更新 Image 控件中的图片，同时利用 UpdateProgress 控件提示更新信息。

(5) 添加一个网页，定时局部更新 Image 控件中的图片。

(6) 建立母版页和内容页，要求在内容页中每 2 秒钟局部更新 Label 控件的当前时间。

(7) 添加一个网页，在两个 UpdatePanel 控件中分别放置一个显示时间的 Label 控件，当单击 UpdatePanel 外面的 Button 控件时，只有其中一个 UpdatePanel 控件局部刷新。

(8) 添加一个网页，使用 SlideShowExtender 扩展控件，自动播放 Image 控件中的图片。

(9) 添加一个网页，只允许文本编辑框输入小写字母。

(10) 添加一个网页，在网页的两边悬浮固定位置的广告。

第10章　Web 服 务

伴随着计算机软硬件技术和 Internet 的快速发展，人们的学习、工作、科研、娱乐越来越依赖网络资源，在应用程序中如何更好地整合和充分、快捷、有效地利用 Internet 上的海量计算能力和海量存储信息，实现在因特网上建立广泛交互服务的“五个任何”(任何人，任何地方，任何时间，任何设备，任何手段)互联梦想，使 Internet 成为一个透明的超级计算机。这个迷人的梦想的实现者就是以 XML(eXtensible Markup Language)或 JSON(JavaScript Object Notation)为基础的数据交换格式、支持 AJAX(Asynchronous JavaScript and XML)的 Web Services(Web 服务)。

支持 AJAX 的 Web 服务是目前最新的技术，它可以轻松整合不同的应用程序以及异构系统之间的数据共享。

本章首先介绍 Web 服务的基本概念和特点，然后通过具体实例说明如何创建和使用支持 AJAX 的 Web 服务。

本章的学习目标

- 了解什么时候需要使用 Web 服务
- 掌握如何创建和调用支持 AJAX 的 Web 服务

10.1　支持 AJAX 的 Web Services 简介

ASP.NET AJAX 提供了完整的架构以从客户端 JavaScript 调用 ASP.NET Web Services。设计者可以轻松地用 AJAX 把服务器端的数据和功能集成到用户响应的 Web 页面中，而所需要做的就是仅仅用[ScriptService]属性来标识 Web Service。ASP.NET AJAX 框架会为 Web Service 自动生成 JavaScript 代理，然后通过使用代理来调用 Web methods。

10.1.1　什么是 Web Services

简单地讲，Web Services(Web 服务)是一个基于因特网的可通过 Web 被远程调用的应用程序模块(API)，比如网站里想提供天气预报的服务，不用自己实现天气预报的功能，只用调用其他公司提供的免费或付费 Web Services 即可。

- 服务就是一个软件，它和客户端应用程序没有很紧密地耦合或关联。服务是可以被动态地发现及组合成其他软件的软件实体。

- Web 服务是一种基于 XML、JSON、SOAP、HTTP、UDDI、WSDL 等一系列标准实现的分布式计算技术和软件组件。
- Web 服务提供了一个松耦合和跨平台的分布式计算环境，它是一个与操作系统无关、程序设计语言无关、机器类型无关、运行环境无关的平台，实现网络上应用的共享，并可用于复杂的系统集成。
- 国际标准化组织 W3C：Web Service 是一个通过 URL 识别的软件应用程序，其界面及绑定能用 XML 文档来定义、描述和发现，使用基于 Internet 协议上的消息传递方式与其他应用程序进行直接交互。
- Microsoft：Web Service 是为其他应用提供数据和服务的应用逻辑单元，应用程序通过标准的 Web 协议和数据格式获得 Web Service，如 HTTP、XML/JSON 和 SOAP 等，每个 Web Service 的实现是完全独立的。Web Service 具有基于组件的开发和 Web 开发两者的优点，是 Microsoft 的.Net 程序设计模式的核心。

Web Service 是一种革命性的分布式计算技术，它使用基于 XML/JSON 的消息处理作为基本的数据通讯方式，消除使用不同组件模型、操作系统和编程语言的系统之间存在的差异，使异类系统能够作为计算网络的一部分协同运行。开发人员可以使用像过去创建分布式应用程序时使用组件的方式，创建由各种来源的 Web 服务组合在一起的应用程序。由于 Web 服务是建立在一些通用协议的基础上，如 HTTP、SOAP、XML、JSON、WSDL、UDDI 等，这些协议在涉及到操作系统、对象模型和编程语言的选择时，没有任何倾向，因此 Web 服务将会有很强的生命力。

SOAP，即 Simple Object Access Protocol，简单对象访问协议。SOAP 是消息传递的协议，它规定了 Web Services 之间是怎样传递信息的。简单地说，SOAP 规定了：

(1) 传递信息的格式为 XML。这就使 Web Services 能够在任何平台上，用任何语言进行实现。

(2) 远程对象方法调用的格式。规定了怎样表示被调用对象以及调用的方法名称和参数类型等。

(3) 参数类型和 XML 格式之间的映射。这是因为，被调用的方法有时候需要传递一个复杂的参数，例如，一个 Person 对象。怎样用 XML 来表示一个对象参数，也是 SOAP 所定义的范围。

WSDL，即 Web Services Description Language，Web 服务描述语言。WSDL 是 Web Services 的定义语言。当实现了某种服务的时候(如：股票查询服务)，为了让别的程序调用，必须告诉大家服务接口。例如：服务名称，服务所在的机器名称，监听端口号，传递参数的类型，个数和顺序，返回结果的类型等等。这样别的应用程序才能调用该服务。WSDL 协议就是规定了有关 Web Services 描述的标准。

UDDI，即 Universal Description Discovery and Integration，通用描述发现和集成协议。简单说，UDDI 用于集中存放和查找 WSDL 描述文件，起着目录服务器的作用。

XML，即 eXtensible Markup Language，可扩展标记语言，是 Internet 上数据表示和数据交换的新标准。它是 ISO(International Organization for Standardization，国际标准化组织)的

SGML(Standard for General Markup Language，通用标记语言标准)的一个简化子集。XML 关注信息本身，是 Web 上表示结构化信息的一种标准文本格式。与传统的注重页面信息显示的 HTML(Hypertext Markup Language， 超文本链接标示语言)相比，关注于内容的 XML 具有以下诸多优点：良好的可扩展性，语言简单有效，可自行定义标记；内容与形式的分离，主要刻画数据内容，不考虑显示效果；有严格的语法要求，便于分析统一和与数据库信息转换；便于传输，为纯文本形式，可通过 Http 协议直接传输，可跨越防火墙；等等。XML 的出现和发展对于 Internet 和 Intranet 产生了巨大的影响。

1. Web 服务的目标

(1) 让地理上分布在不同区域的计算机和设备一起工作，为用户提供各种各样的服务。

(2) 用户可以控制要获取信息的内容、时间、方式，而不必像目前这样，在无数个信息孤岛中浏览，去寻找自己需要的信息。

2. Web 服务的影响

(1) Web 服务支持在 Web 站点上放置可编程的元素，用户可以抓取已有的元素，构成自己的新服务。

(2) 能进行基于 Web 的分布式计算和处理，能很好地兼容现有的 Web 技术。

(3) Web 服务使得 Internet 成为一个可以无限扩展、拥有无限潜力的分布式计算平台。

(4) 任何设备可以随时随地访问 Internet 上的 Web 服务。

(5) 软件模块充分复用、计算机资源充分共享、信息无缝共享和交流。

3. Web 服务的好处

(1) Web 服务改变企业对 IT 系统的应用方式和依赖程度。在 Web 服务的帮助下，企业可以不再耗费巨资建设 IT 系统，转而使用 Internet 上提供的各类 Web 服务来实现。

(2) 利用 Web 服务，公司和个人将能够迅速且廉价地向整个国际互联网络提供他们的服务，进而建立全球范围的联系，在广泛的范围内寻找可能的合作伙伴。

4. Web 服务的主要特征

(1) 互操作性：一个 Web 服务可与其他 Web 服务交互，协同工作；可以使用任何语言开发 Web 服务或使用他人提供的 Web 服务；开发环境可以异构。

(2) 普遍性：Web 服务使用 HTTP 和 XML 进行通信，支持这些技术的设备都可以拥有和访问 Web 服务。

(3) 松散耦合：Web 服务的实现对使用者透明，当服务的实现发生变动时不影响用户使用。

(4) 高度可集成能力：Web 服务和采用了简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范，屏蔽了平台的异构性，CORBA、DCOM、EJB 等都可通过它进行互操作。

5. Web 服务架构

SOA(Service Oriented Architecture)架构，如图 10-1 所示。

3 个参与者：服务提供者、服务请求者、服务中介者。

3 个基本操作：发布、发现、绑定。

XML 或 JSON 是数据的格式，SOAP 是调用 Web Services 的协议，WSDL 是描述 Web Services 的格式，而 UDDI 是 Web Services 发布，查找和利用的组合。

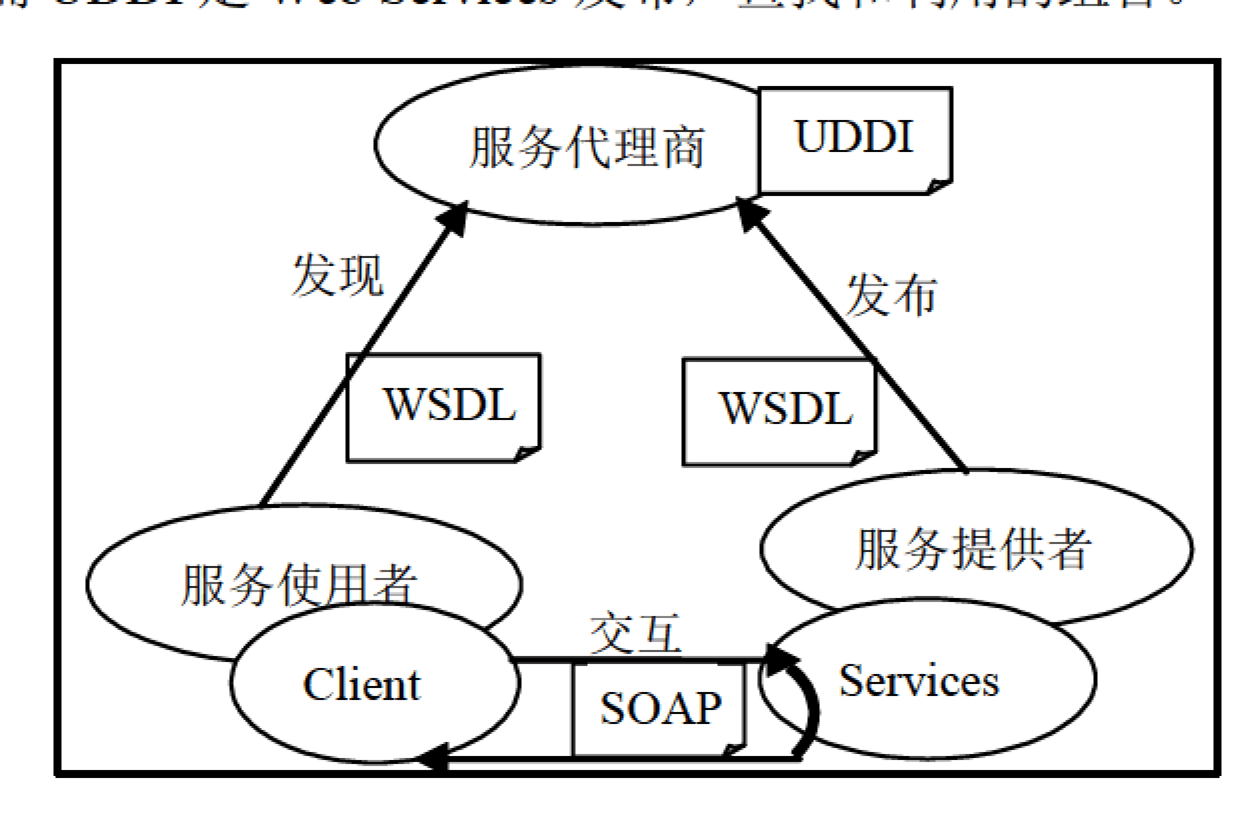


图 10-1　SOA Web 服务架构

6. Web 服务有两种类型的应用

(1) 可重复使用的应用程序组件

有一些功能是不同的应用程序常常会用到的。那么为什么要周而复始地开发它们呢？

Web services 可以把应用程序组件作为服务来提供，比如汇率转换、天气预报或者甚至是语言翻译等等。

比较理想的情况是，每种应用程序组件只有一个最优秀的版本，这样任何人都可以在其应用程序中使用它。

(2) 连接现有的软件

通过为不同的应用程序提供一种链接其数据的途径，Web services 有助于解决协同工作的问题。

通过使用 Web services，用户可以在不同的应用程序与平台之间来交换数据。

10.1.2　JSON 和支持 AJAX 的 Web 服务

JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，易于阅读和编写，易于机器解析和生成。它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 的一个子集。JSON 采用完全独立于语言的文本格式，使用了类似于 C 语言家族的习惯(包括 C, C++, C#, Java, JavaScript, Perl, Python 等)。这些特性使 JSON 成为理想的数据交换语言。

JSON 建构于两种结构：

(1) “名称/值”对的集合(A collection of name/value pairs)。不同的语言中，它被理解为

对象(object)，纪录(record)，结构(struct)，字典(dictionary)，哈希表(hash table)，有键列表(keyed list)，或者关联数组(associative array)。

(2) 值的有序列表(An ordered list of values)。在大部分语言中，它被理解为数组(array)。

这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON 以一种特定的字符串形式来表示 JavaScript 对象。如果将具有这样一种形式的字符串赋给任意一个 JavaScript 变量，那么该变量会变成一个对象引用，而这个对象就是字符串所构建出来的，好像有点拗口，还是用实例来说明。

这里假设需要创建一个 User 对象，并具有用户 ID、用户名、用户 Email 这 3 个属性，可以使用以下 JSON 形式来表示 User 对象：

```
{"UserID":110, "Name":"Sunny", "Email":happy@gmail.com"};
```

然后如果把这一字符串赋予一个 JavaScript 变量，那么就可以直接使用对象的任一属性了。完整代码:

```
<script>var User = {"UserID":110, "Name":"Sunny", "Email":":happy@gmail.com"};
alert(User.Name); </script>
```

借助 ASP.NET AJAX Extension，微软选择 JSON 在服务器和 Ajax 客户端实现数据交换，从而创建支持 AJAX 的 Web 服务。在客户端和服务器端都实现了(数据的)串行化器和并行化器以使数据按 JSON 的格式交换。网页中的客户端脚本与服务器通过 Web 服务通信层进行通信来访问 Web 服务，该通信层使用 AJAX 技术进行 Web 服务调用，数据在客户端与服务器之间通常采用 JSON 格式进行异步交换。

通常，默认将 JavaScript 对象序列化为 JSON 格式，使用 JavaScript eval 函数可进行反序列化。但 Web 服务和 ASP.NET 网页中的单个方法可以返回其他格式(如 XML)。可以通过 ScriptMethod 属性来指定方法的序列化格式。对于某个 ASMX 服务，可以设置 ScriptMethod 属性形如[ScriptMethod(ResponseFormat = ResponseFormat.Xml)]以使某个 Web 服务方法返回 XML 数据。

10.2　创建和使用支持 AJAX 的 Web 服务

使用 VWD 2008 可以创建能够从客户端脚本访问的 ASP.NET 自定义 Web 服务(.asmx 文件)。在【例 10-1】中，将创建一个支持 AJAX 的 Web 服务，并使用客户端脚本调用其方法。

【例 10-1】在 VWD 2008 中创建支持 AJAX 的 Web 服务。要完成本示例，需要在本地计算机上安装 Microsoft Internet 信息服务(IIS)。

(1) 创建支持 AJAX 的 Web 服务的网站。说明：在本实例中必须使用 IIS 网站。

① 打开 VWD 2008。

② 在【文件】菜单上单击【新建网站】，出现【新建网站】对话框。

③ 在【Visual Studio 已安装的模板】下单击【ASP.NET 网站】，单击【浏览】，单击【本地 IIS】，单击【默认网站】，在右上角单击【创建新 Web 应用程序】图标，如图 10-2 所示。VWD 2008 创建了一个新的 IIS Web 应用程序。



图 10-2　【选择位置】对话框

指定名称 AJAXsupportedWebService，不选择【使用安全套接字层(U)】。

单击【打开】，出现【新建网站】对话框，新网站的名称位于【位置】列表中，包括协议 http://和位置 localhost，形如 http://localhost/AJAXsupportedWebService(这指示处理本地 IIS 网站)。在【语言】列表中，选择编程语言 Visual C#，单击【确定】。

(2) 编写支持 AJAX 的自定义 Web 服务

① 在【解决方案资源管理器】中，右击网站的名称，单击【添加新项】。

② 在【Visual Studio 已安装的模板】下，单击【Web 服务】，然后在【名称】文本框中输入 GreetingsWebService.asmx，如图 10-3 所示。

③ 确保【将代码放在单独的文件中】复选框已选定，然后单击【添加】。

VWD 2008 创建一个新的 Web 服务，它由两个文件组成。GreetingsWebService.asmx 文件是用来调用 Web 服务方法的文件，它指向 Web 服务代码。代码本身位于 App_Code 文件夹中的类文件(GreetingsWebService.cs)中，代码文件包含 Web 服务的模板。

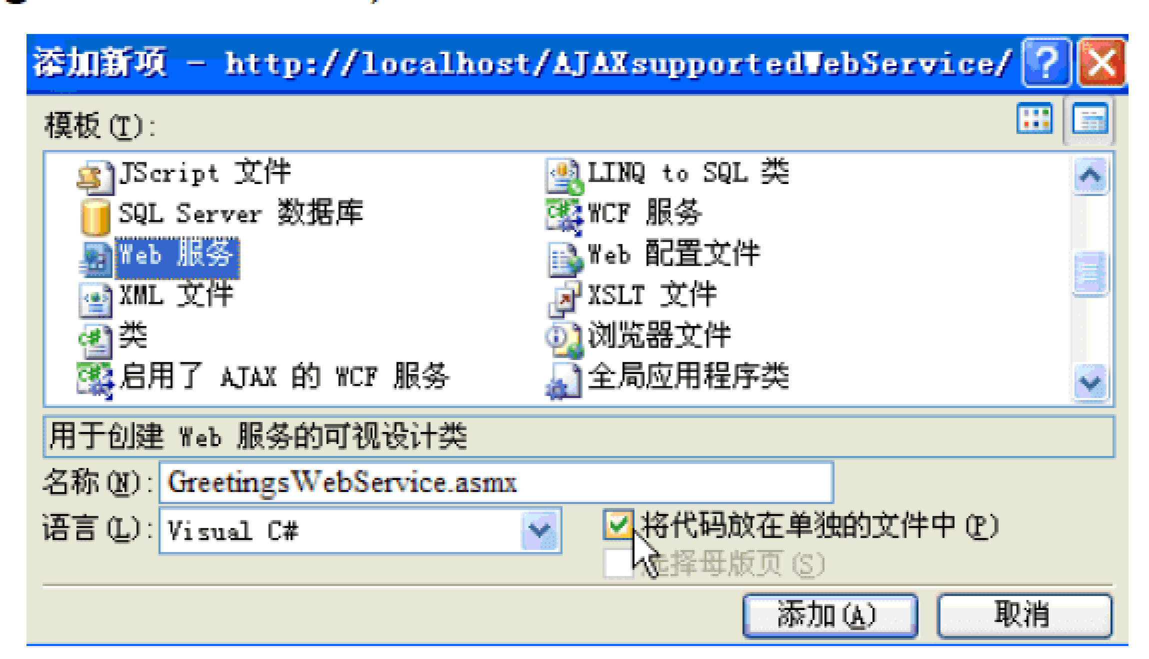


图 10-3　【添加新项】对话框

④ 打开文件 GreetingsWebService.asmx，修改为如下所示的代码。

```
<%@ WebService Language="C#" CodeBehind="~/App_Code/GreetingsWebService.cs"
        Class=" Samples.Aspnet.GreetingsWebService" %>
```

其中，@WebService 指令表示在该文件中声明一项 Web 服务，Class 属性指明实现该

Web 服务的类的类名是 GreetingsWebService，Language 属性指明使用的语言是 C#，CodeBehind 属性指明实现该服务的代码文件是~/App_Code/GreetingsWebService.cs。

⑤ 打开文件 GreetingsWebService.cs，可看到如下所示的代码。并将 1~20 行代码用 namespace Samples.Aspnet 名称空间命令括起来。

```
   namespace Samples.Aspnet
   {
1   /// <summary>
2   ///GreetingsWebService 的摘要说明
3   /// </summary>
4   [WebService(Namespace = "http://tempuri.org/")]
5   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1) ]
6   //若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。
7   [System.Web.Script.Services.ScriptService]
8   public class GreetingsWebService : System.Web.Services.WebService
9   {
10       public GreetingsWebService ()//类的构造函数
11       {
12           //如果使用设计的组件，请取消注释以下行
13           //InitializeComponent();
14       }
15       [WebMethod]
16       public string HelloWorld()
17       {
18           return "Hello World";
19       }
20   }
21   }
```

代码的 8~10 行定义了一个名为 GreetingsWebService 的类，从 WebService 类派生。使用 ASP.NET 创建的 Web 服务的类可以选择从 WebService 类派生，以获得对通用 ASP.NET 对象的访问权限，例如 Application、Session、User 和 Context。

代码的第 4 行是应用 WebService 属性设置 Web 服务所属的 XML 命名空间。

将可选的 WebService 属性应用于实现 Web 服务的类，以便为该 Web 服务设置默认的 XML 命名空间(该命名空间原来是 http://tempuri.org)，同时还用一个字符串来描述该 Web 服务。

在使该 XML Web services 可以公开使用之前，强烈建议修改此默认命名空间 http://tempuri.org。这一点很重要，因为必须将该 XML Web services 与其他可能不慎将该命名空间用作默认命名空间(<http://tempuri.org/>)的 XMLWeb services 相区分。

代码的 15-19 行定义 Web 服务方法。实现 Web 服务的类的方法不会自动具有通过 Web 进行通信的能力，但通过使用 ASP.NET 创建的 Web 服务，很容易添加该功能。若要添加此功能，请将 WebMethod 属性应用于公共方法。可通过 Web 进行通信的 Web 服务的方法称为

Web 服务方法。

请注意，函数名 HelloWorld 前面放置了[WebMethod]属性，作为函数声明的一部分。此外，类 HelloWorld 由[ScriptService]属性限定。利用这些属性，可以从支持 AJAX 的 ASP.NET 网页中的脚本调用 Web 服务。

(3) 测试 Web 服务。现在就可以在浏览器中测试 Web 服务，此测试不使用脚本来调用 Web 服务方法。测试的目的仅仅是验证 Web 服务是否正常运行。

① 打开浏览器，输入 URL：

```
http://localhost/AJAXsupportedWebService/ GreetingsWebService.asmx.asmx
```

来调用 Web 服务，在浏览器中会显示一个页，显示由 Web 服务公开的方法。

或：确保当前文件是 GreetingsWebService.asmx，按 CTRL+F5，可以在浏览器中看到该程序运行的界面如图 10-4 所示。

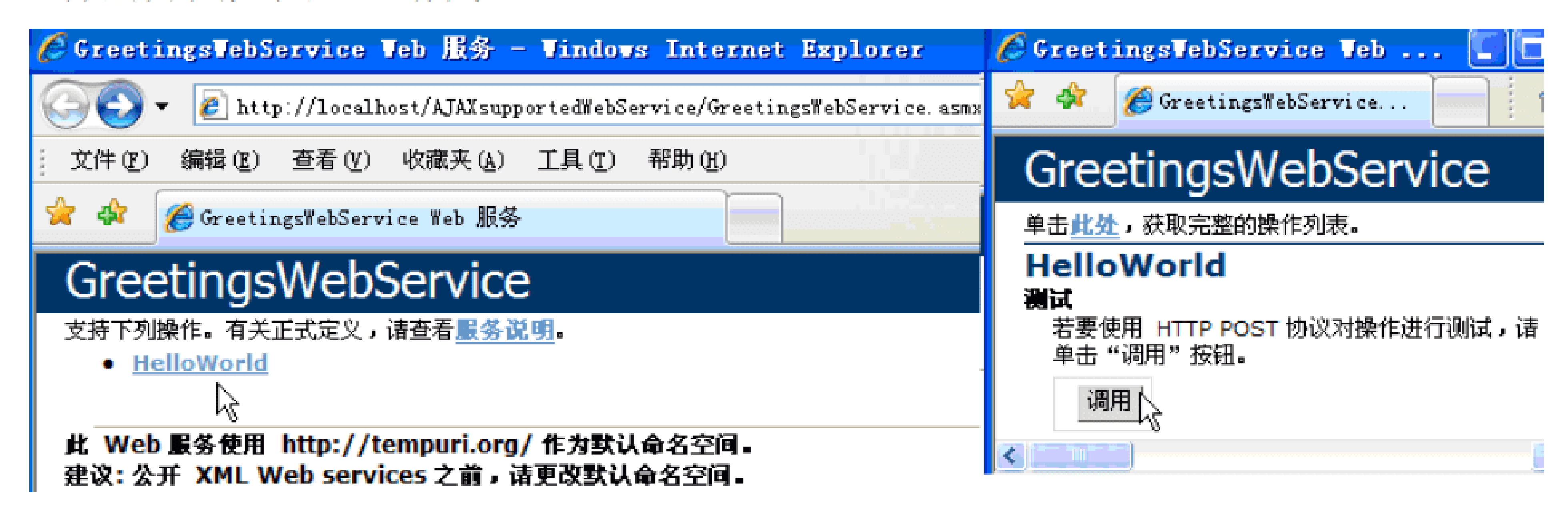


图 10-4　GreetingsWebService.asmx 运行示意图和调用 Web 服务 Hello 的界面

② 单击 HelloWorld。出现一个带有“调用”(Invoke)按钮的页。

③ 单击“调用”按钮调用方法，可看到如图 10-5 所示的 XML 格式的结果。

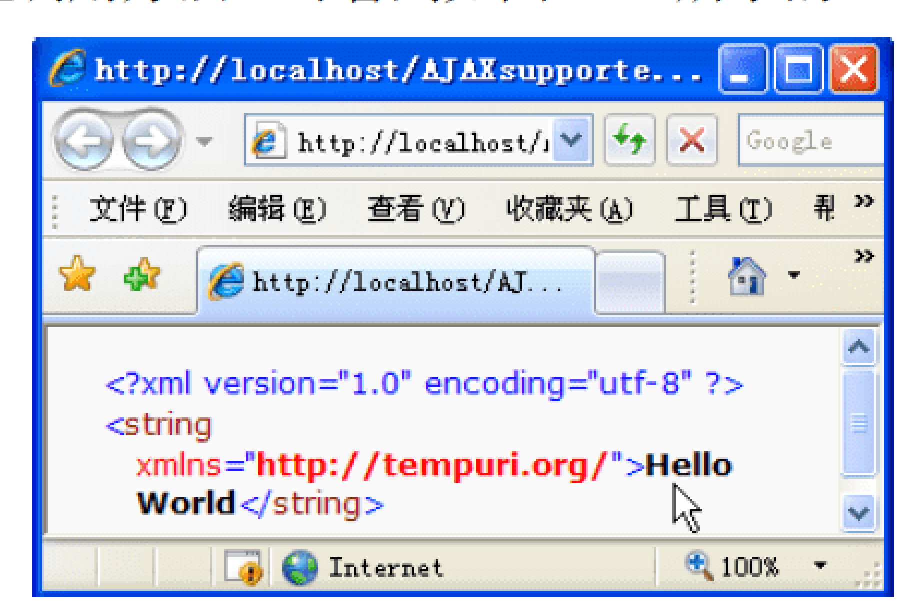


图 10-5　HelloWorld 服务的返回结果

④ 单击图 10-4 中的【服务说明】链接，将看到用 WSDL 语言描述该 Web 服务的 XML 文档。

⑤ 关闭浏览器。保存文件并将其关闭。已完成创建支持 AJAX 的 Web 服务。

(4) 打开文件 GreetingsWebService.cs 添加如下的 Web 服务方法。

```
[WebMethod(Description = "无参数的 Web 服务，显示“Hello world！和服务器的当前时间”" )]
public string Greetings()
{
```

```
        string serverTime = String.Format("Current date and time: {0}.", DateTime.Now);
        string greetings = "Hello World! <br/>" + serverTime;
        return greetings;
    }
    [WebMethod(Description = "有参数的 Web 服务，显示“Hello world 您输入的用户名！"
        +"和服务器的当前时间” ")]
    public string GreetingsParamters(string vistorName)
    {
        string inputString = Server.HtmlEncode(vistorName);
        if (!String.IsNullOrEmpty(inputString)) //if (vistorName.Trim().Length != 0)//also ok.
        {
            string serverTime =String.Format("Current date and time is {0}.", DateTime.Now);
            string greetings = "Hello, " + inputString + "!<br/>" + serverTime;
            return greetings;
        }
        else return "my friend,please enter your name!";
    }
```

(5) 从客户端脚本调用 Web 服务。

① 在【解决方案资源管理器】中，右击网站的名称(http://localhost/HelloWorldService)，然后单击【添加新项】。

② 在【VWD 2008 已安装的模板】下，单击【JScript 文件】，然后在【名称】文本框中输入 callGreetingsClientJScript.js，单击【确定】。

③ 向脚本文件中添加以下代码：

```
    var GreetingsServiceProxy;// Initializes global and proxy default variables.
    function pageLoad()
    {
        // Instantiate the service proxy.
        GreetingsServiceProxy = new   Samples.Aspnet.GreetingsWebService();
        // Set the default call back functions.
        GreetingsServiceProxy.set_defaultSucceededCallback(SucceededCallback);
        GreetingsServiceProxy.set_defaultFailedCallback(FailedCallback);
        GreetingsServiceProxy.set_timeout(2000);
        GreetingsServiceProxy.set_defaultUserContext("my context");
    }
    // Processes the button click and calls the service Greetings method.处理按钮事件，调用 Web 服务
    function OnClickGreetings()
    {
        var greetings =   GreetingsServiceProxy.Greetings();
    }
    // Processes the button click and calls the service Greetings method with parameters.
    function OnClickGreetingsParameters(s) //处理按钮事件，调用带参数的 Web 服务
    {
```

```
    var greetings =GreetingsServiceProxy.GreetingsParamters(s,SucceededCallback,FailedCallback);
}
function SucceededCallback(result) // Callback function that processes the service return value.
{
    var RsltElem = document.getElementById("Results");
    RsltElem.innerHTML = result;
}
// Callback function invoked when a call to the service methods fails.
function FailedCallback(error, userContext, methodName)
{
    if(error !== null)
    {
        var RsltElem = document.getElementById("Results");
        RsltElem.innerHTML = "An error occurred: " + error.get_message();
    }
}
if (typeof(Sys) !== "undefined") Sys.Application.notifyScriptLoaded();
```

在调用某个 Web 服务时，通常定义两个回调：一个用于成功的情况，另一个用于失败的情况。Web 服务调用是异步执行的，必须通过回调来了解调用的实际完成情况。这里的方法 SucceededCallback 和 FailedCallback 分别实现成功回调和失败回调。

callGreetingsService.aspx 页面将使用此客户端脚本来调用 GreetingsWebService.cs 中定义的 Greetings( )和 GreetingsParamters( )这 2 个 Web 服务方法。

④ 切换到 callGreetingsService.aspx 的【设计】视图，从工具箱的【AJAX Extensions】组中向页面中拖放一个 ScriptManager 控件，使用默认的控件名称，然后设置 ScriptManager 控件的 Services 属性。需要从客户端代码中用 ScriptManager 注册任何想要与之交互的服务。单击 Services 属性右侧的【浏览】按钮，打开【ServiceReference 集合编辑器】，单击【添加】按钮，在右侧设置 path 为 GreetingsWebService.asmx，即完成向 ScriptManager 控件中添加服务引用，如图 10-6 所示。同理，为 Scripts 属性添加对脚本文件的引用路径 path 为 callGreetingsClientJScript.js。



图 10-6　为 ScriptManager 控件添加属性

⑤ 在 body 的 form 标记中，可看到如下所示的代码：

```
<asp:ScriptManager runat="server" ID="scriptManager">
```

```
        <Services> <asp:ServiceReference Path="GreetingsWebService.asmx" /> </Services>
        <Scripts> <asp:ScriptReference Path="callGreetingsClientJScript.js" /> </Scripts>
    </asp:ScriptManager>
```

当然，可在源视图中直接输入上述 ScriptManager 控件的代码，这同样可完成上一步的操作。

该页包含 ScriptManager 控件。Services 节的 ServiceReference 元素中的 path 属性引用先前生成的 GreetingsWebService.asmx 服务。Script 节的 ServiceReference 元素中的 path 属性引用 callGreetingsClientJScript.js 脚本。

ScriptManager 控件包括在 ASP.NET 3.5 AJAX Extensions 中，它用来处理页面上的所有组件以及页面局部更新，生成相关的客户端代理脚本以便能够在 JavaScript 中访问 Web Service，所有需要支持 ASP.NET AJAX 的 ASP.NET 页面上有且只能有一个 ScriptManager 控件。在 ScriptManager 控件中可以指定需要的脚本库，或者指定通过 JS 来调用的 Web Service，以及调用 AuthenticationService 和 ProfileService，还有页面错误处理等。

ASP.NET Web 服务通过 ScriptManager 控件向客户端脚本公开之后，ASP.NET 将为这些 Web 服务自动创建 JavaScript 代理类。可以通过调用 JavaScript 代理类的相应方法来调用 Web 服务方法。

⑥ 在 body 的 form 标记中，继续添加如下所示的代码：

```
<span class="style1">记着要将 GreetingsWebService.cs 文件中的 Web 服务类的属性
   [System.Web.Script.Services.ScriptService]启用，即不要注释它。</span>
<hr />从客户端脚本调用：
<table>
  <tr align="left">
    <td>Click to call the Greetings service:</td>
    <td><%--调用回调函数的控件是 HTML 控件--%>
        <input id="ButtonGreet" type="button"
        onclick ="OnClickGreetings(); return false;" value="Greetings" />
    </td>
  </tr>
</table><hr />
```

这里将客户端 HTML 控件 Input(Button)的 onclick 属性和 callGreetingsClientJScript.js 客户端脚本文件中的函数 OnClickGreetings()绑定，在引发该 Input(Button)控件的 Click 事件时将调用并执行客户端脚本文件中的 OnClickGreetings 函数。

这里用的 id="ButtonGreet"按钮是【工具箱】的 HTML 组里的 Input(Button)，这是客户端控件，不同于服务器端控件 Button(【标准】组中的)。

⑦ 在浏览器中测试 Web 服务。打开浏览器，输入以下 URL：

```
http://localhost/AJAXsupportedWebService/callGreetingsService.aspx
```

⑧ 在显示的页中，单击 Greetings 按钮。这将生成对该 Web 服务的调用，这次调用将返

回一条问候消息以及当前服务器日期和时间。

⑨ 切换到 callGreetingsService.aspx 的源视图，在 body 的 form 标记中，继续添加如下所示的代码：

```
<hr />从客户端脚本调用：<br />输入用户名：
<asp:TextBox ID="TextBoxParameters"   runat="server"></asp:TextBox>
<asp:Button ID="ButtonGreetParam"   OnClientClick ="GreetingsParameters();
        return false;"   runat="server" Text="GreetingsParameters()" /><hr />
```

将下列 javascript 客户端脚本块添加到<head></head>标记后。

```
<script type ="text/javascript"   language ="javascript"   >
  function   GreetingsParameters()
  {
      window.alert("now calling GreetingsParameters ()....");
      //获取文本框 TextBoxParameters 服务器控件的值
      var s=document.getElementById ("TextBoxParameters").value;
      window.alert(s);
      //调用客户端脚本文件 callGreetingsClientJScript.js 中的/函数 OnClickGreetingsParameters();
      OnClickGreetingsParameters(s);
   }
</script>
```

这里将服务器端控件 Button 的 OnClientClick 属性和 JavaScript 客户端脚本块中的函数 GreetingsParameters()绑定，在引发该 Button 控件的 Click 事件时将调用并执行客户端脚本中的 GreetingsParameters 函数。

⑩ 按 CTRL+F5 测试运行结果。

(6) 在服务器端使用 C#代码直接调用 Web 服务。

① 切换到 callGreetingsService.aspx 的源视图，在 body 的 form 标记中，继续添加如下所示的代码：

```
<hr />从服务器端调用<br />请输入用户名
<asp:TextBox ID="TextBoxName" runat="server" ForeColor="#CCCCCC">孙悟空
  </asp:TextBox>
<asp:Button ID="ButtonServerSide" runat="server"    Text="从服务端调用 Greetings Web 服务"
  onclick="ButtonServerSide_Click" />
<asp:Label ID="LabelGreetings" runat="server" Text="Label"></asp:Label><hr/>
```

② 在 callGreetingsService.aspx.cs 文件中加入下列代码。

```
Samples.Aspnet.GreetingsWebService myGreetingsService =
        new Samples.Aspnet.GreetingsWebService();
protected void ButtonServerSide_Click(object sender, EventArgs e)
{
```

```
//使用下面的代码可确定是否已为单个页启用 AJAX 功能。
ScriptManager sm = ScriptManager.GetCurrent(Page);
if (sm == null)
{
  //确定是否已为本网页启用 ASP.NET AJAX 功能
  // ASP.NET AJAX functionality is not enabled for the page.
  string str = "alert('" + "ASP.NET AJAX functionality is not enabled for the page." + "')";
  Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "Warning", str, true);
}
else
{
  // AJAX functionality is enabled for the page.
  LabelGreetings.Text =myGreetingsService.GreetingsParamters(TextBoxName.Text.ToString());
  string str = "alert('" + "ASP.NET AJAX functionality is   enabled for the page." + "')";
  Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "Warning", str, true);
}
}
```

③ 切换到 callGreetingsService.aspx 的源视图，按 Ctrl+F5，测试调用结果。

10.3　支持 AJAX 的 Web 服务的应用程序体系结构

典型的 AJAX 体系结构相当容易理解。图 10-7 展示了其工作原理的高层视图。其中有一个由应用程序特定服务组成的后端，通常只可调用 AJAX 脚本的外层，其下方是业务逻辑所在和发挥作用的系统中间层。服务与前端通过 HTTP 交换数据，使用多种格式传递参数和返回值。前端由运行于客户端上的 JavaScript 代码组成，在接收和处理完数据后，它面临着使用 HTML 和 JavaScript 构建图形用户界面的重大任务。对 JavaScript 的依赖是由于受浏览器结构的限制，只有当浏览器可以支持功能更加强大的编程功能时，这种情况才会改变。

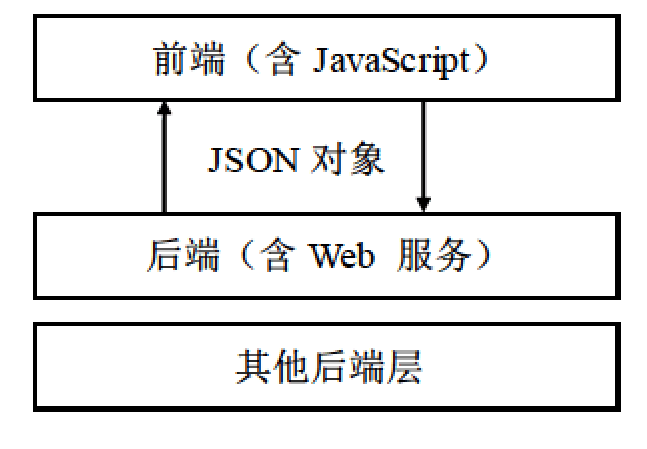


图 10-7　典型的 AJAX 体系结构

在 AJAX 中，服务表示驻留在应用程序域并向客户端脚本代码公开功能的一段代码。

最适合 AJAX 应用程序的服务主要涉及向 Web 客户端公开数据和资源。它可以通过 HTTP 获得，并要求客户端使用 URL(也可以是 HTTP 头)访问数据和命令操作。客户端与服务进行交互使用的是 HTTP 动词，如 GET、POST、PUT 和 DELETE。换句话说，URL 代表

一个资源，而 HTTP 动词描述了想对资源采取的操作。在这些交互中交换的数据以简单格式表示，如 JSON 和纯 XML，也可以聚合格式表示，如 RSS 和 ATOM。

这里用“AJAX 服务”表示通过 JavaScript 脚本服务方法实现的 AJAX 应用程序后端的服务。

10.3.1　AJAX 服务返回什么？

公开 AJAX 服务的唯一方式是通过 HTTP，几乎可以使用任何文本格式来包装请求和响应的主体。JavaScript Object Notation(JSON)是最常用的格式，但也可使用其他格式，如纯 XML 和原始文本。

JSON 是基于文本的格式，用于表示跨应用程序的各层的移动对象的状态。JSON 字符串通过常见的 eval 函数便可方便地赋给 JavaScript 对象。JSON 格式描述了对象，如下所示：

```
{"ID":"xunlei", "Company":"迅雷公司"}
```

该字符串表示一个对象有两个属性，即 ID 和 Company，以及它们各自的文本序列化值。如果对某个属性赋予非基本类型的值(比如自定义对象)，那么该值将递归地序列化为 JSON，如下所示：

```
{"ID":"xunlei","Company":"迅雷公司","Location": "{"City":"深圳", "Country":"中国"}",}
```

使用 eval 函数进行处理时，JSON 字符串将变成一个关联性数组(即一种名称/值的集合)，其中每个条目都有一个名称和值。如果 JSON 字符串用于代表一个自定义对象(比如 Customer)的状态，那么，必须负责确保客户端具有相应类的定义。换句话说，JavaScript 的 eval 函数只是将 JSON 字符串中的信息提取到一个通用容器。如果需要将此信息公开为一个自定义对象(比如 Customer 对象)，那么提供类定义并将数据载入到其中的任务就完全依靠设计者或设计者使用的框架来完成。

10.3.2　JSON 与 XML 之比较

多年以来，XML 已被推崇为 Web 通用语言。众所周知，现在 AJAX 正在改变着 Web，就数据表示而言，JSON 更受欢迎，而 XML 正在被推向角落。

JSON 稍微简单些，更适合与 JavaScript 语言配合使用。读者可能会争辩哪一个更容易为人们所理解，不过，Web 浏览器处理 JSON 肯定比处理 XML 更容易。使用 JSON 时，不需要 XML 分析器之类的任何东西。为分析文本所需的一切都已经完全构建在 JavaScript 语言中了。

凭借 JSON，还能以相对较低的成本在体系结构方面赢得关键优势。可以按照对象无处不在的思路进行推理。在服务器上可定义一些实体，并用最喜爱的托管语言将它们实现为类。当某个 Web 服务方法需要返回任何类的一个实例时，该对象的状态被序列化为 JSON 并通过线路传送。客户端接收并处理 JSON 字符串，并将其内容载入一个数组或一种与服务器类有

相同接口的镜像 JavaScript 对象。类的接口是从 JSON 流推断出来的。这样，服务和客户端页面的代码便使用一个实体的同一逻辑定义。

单纯从技术角度来说，AJAX 服务并不严格要求 JSON 实现为数据表示格式。使用 XML 也可实现同样的结果。

10.3.3　ASP.NET AJAX 中的服务

通过 ASP.NET AJAX Extensions 实现脚本服务有两种方式——使用特殊类型的 ASP.NET Web 服务和通过页面方法。前一种情况下，只需设计和构建一个链接到 ASMX 资源的类：

```
<%@ WebService Language="C#" CodeBehind="~/App_Code/TimeWebService.cs" Class="
IntroAjax.WebServices.TimeWebService" %>
```

该类可以选择从 WebService 类继承，并且必须用新的 ScriptService 属性加以修饰：

```
[ScriptService]
public class TimeWebService : System.Web.Services.WebService
{
  [WebMethod]
  public string GetServerTime()
  {
    string serverTime = String.Format("The current time is {0}.", DateTime.Now);
    return serverTime;
  }
}
```

每个可调用脚本的方法均声明为公共方法，并标记有通常的 WebMethod 属性。

页面方法只不过是在单个 ASP.NET 页面上下文中定义的公共、静态方法，每个方法都标记为 WebMethod。只能从宿主页面中调用它们。除了存储不同之外，对 Web 服务或页面方式的调用均由 ASP.NET AJAX 环境以相同方式进行处理。

AJAX 服务代表后端的一部分，从 WS-* Web 服务意义上讲，它们并不是公共 Web 服务，Web 服务都是通过 WSDL 脚本完整记录并可通过携带 SOAP 数据的 POST 命令进行访问。AJAX 服务是真正的本地服务，通常在调用它们的同一个应用程序中进行定义。然而，它们也可以在不同的 Web 应用程序甚至不同的网站上，前提是它们位于同一个域。

通过启用 ASP.NET AJAX 运行时以接受对服务的调用，ScriptService 属性扮演了一个关键角色。如果没有 ScriptService 属性，当用户试图进行调用时，服务器上就会引发异常。图 10-8 显示当一个 AJAX 页面链接到未标记该属性的服务时所返回的消息。因为 ASP.NET 内部机制拒绝处理缺少 ScriptService 属性的 ASP.NET Web 服务进行的脚本调用。

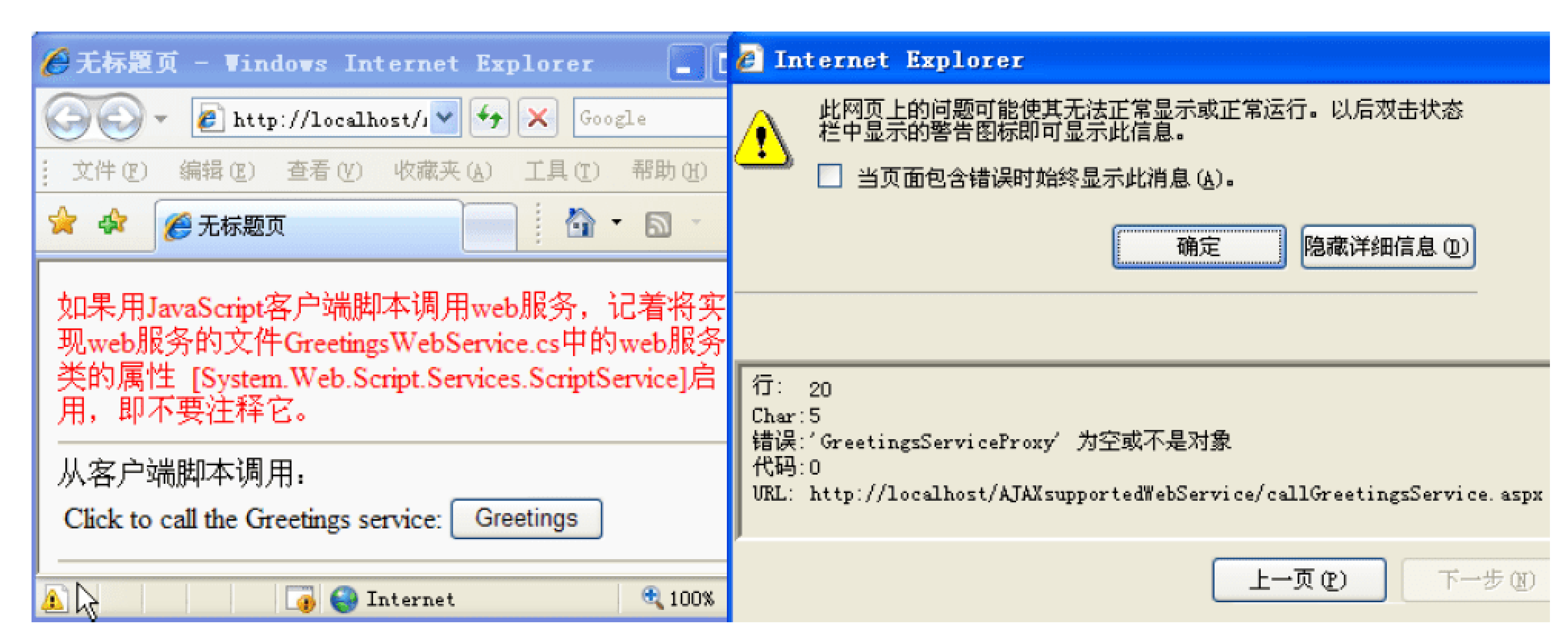


图 10-8　引用不可脚本化服务的页面

默认情况下，调用 AJAX 服务方法的 JavaScript 代理类只能使用 HTTP POST 动词调用并返回 JSON 格式的数据。然而，可以通过给 Web 服务方法添加可选择的 ScriptMethod 属性，使它可以使用 HTTP GET 动词和 XML 格式。表 10-1 详细说明了 ScriptMethod 属性所支持的参数。返回到客户端的数据可以更改为 XML，甚至可以添加对 GET 请求的支持。但这种表面看起来无恶意的更改可能会为攻击者提供新的机会，且增加了跨站点调用该方法的可能性。下列代码显示了一个 Web 服务方法的定义：

```
[WebMethod]
[ScriptMethod]
public string GetServerTime()
{
    string serverTime = String.Format("The current time is {0}.", DateTime.Now);
    return serverTime;
}
```

表 10-1　ScriptMethod 属性的参数

参　数	说　明
ResponseFormat	指定是否将响应序列化为 JSON 或者 XML。默认为 JSON，但是，当方法的返回值是 XmlDocument 时，XML 格式会比较方便
UseHttpGet	表明是否可以使用 HTTP GET 动词调用 Web 服务方法。由于安全性原因，此项的默认设置为 false
XmlSerializeString	表明包括字符串在内的所有返回类型是否都序列化为 XML。默认为 false。当响应格式设置为 JSON 时，将忽略该属性的值

注意：

用 ScriptService 属性修饰的 AJAX 脚本服务仍可由能够访问 ASMX 服务端点且能处理 SOAP 协议的代码访问。如图 10-9 所示。

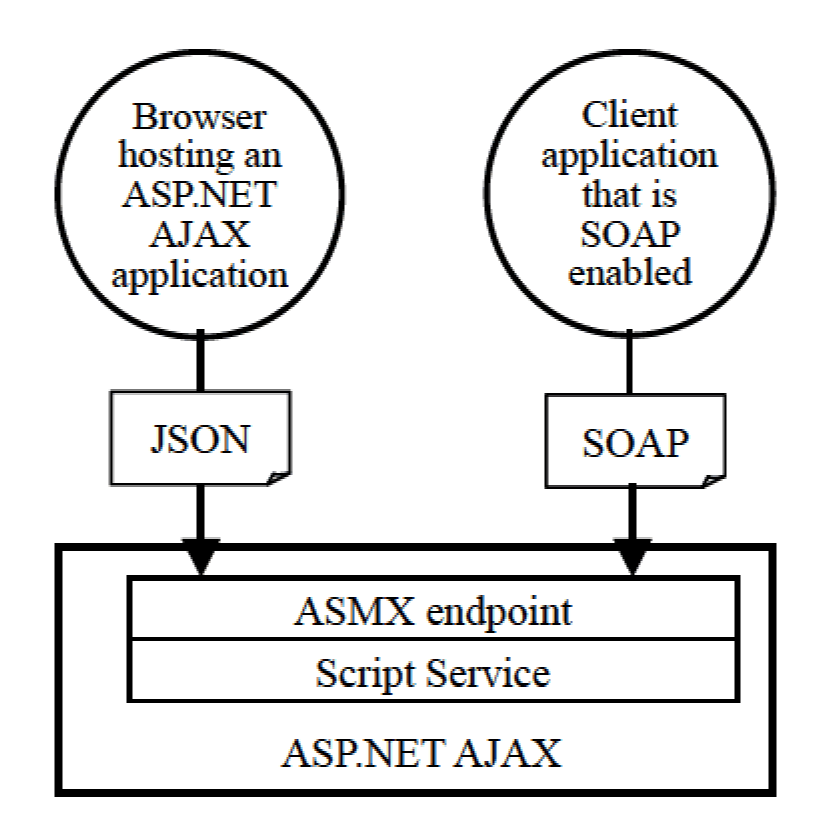


图 10-9　AJAX Web 服务采用 JSON 或 SOAP 通信格式

10.3.4　ASP.NET AJAX 服务和 SOAP

一旦创建了 ASP.NET AJAX 脚本服务，它就发布为 ASMX 资源。默认情况下，它是公共 URL，可以由 AJAX 客户端通过 JSON 格式使用，也可以由 SOAP 客户端及工具发现和使用。ASP.NET AJAX 服务默认情况下具有双重功能：JSON 和 SOAP。不过，可以选择禁用 SOAP 客户端及工具。只需在承载服务的 ASP.NET 应用程序的 Web.config 文件中输入下列配置设置即可关闭 SOAP 客户端：

```
<system.Web>
  <WebServices>
    <protocols>
      <clear />
    </protocols>
  </WebServices>
</system.Web>
```

这个简单的设置禁用了为 ASP.NET 3.5 Web 服务定义的任何协议(特别是 SOAP)，并让服务仅回复 JSON 请求。

注意：

当这些设置开启时，无法再通过浏览器的地址栏调用 Web 服务来进行快速测试。同样，也不能请求 WSDL 向 URL 中添加?wsdl 后缀。

为了在 Web 应用程序中启用 ASP.NET AJAX，必须在 Web.config 文件中包含下列设置：

```
<httpHandlers>
  <remove verb="*" path="*.asmx"/>
  <add verb="*" path="*.asmx" validate="false"
     type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
     Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
  <add verb="*" path="*_AppService.axd" validate="false"
     type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
```

```
        Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    <add verb="GET,HEAD" path="ScriptResource.axd"
        type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
        Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" validate="false"/>
</httpHandlers>
```

<remove>节会丢弃 ASMX 资源的默认 HTTP 处理程序，即通过 SOAP 处理请求的处理程序。<add>节点会添加一个新的 HTTP 处理程序，该处理程序主要检查每个传入的 ASMX 请求的内容类型，如果内容类型的头部设置为 application/json，则通过 JSON 处理该请求。否则，HTTP 处理程序假定该请求是基于 SOAP 的，并将其转发给 ASP.NET 3.5 Web 服务处理程序。如果禁用了 SOAP 协议，则会拒绝该请求。

最后，不管表面如何，没有任何必要让 SOAP 介入 ASP.NET AJAX 服务。但对 SOAP 客户端的支持是得到保证的，除非在 Web.config 文件中明确禁用。

要使 ASP.NET AJAX 服务按预期正常工作，传入请求的内容类型 HTTP 头必须设置为 application/json。对于通过<script>标记实施的跨站点攻击来说，这还是极好的补救措施。

10.3.5　调用 AJAX 服务

(1) 添加调用 AJAX Web 服务的客户端 javascript 脚本文件，如以下代码所示：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
  <Scripts> <asp:ScriptReference Path="~/ScriptPath/TimeServiceJScript.js" /> </Scripts>
</asp:ScriptManager>
```

(2) 添加 Web 服务客户端代理，如以下代码所示：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
   <Services>   <asp:ServiceReference Path="TimeWebService.asmx" /> </Services>
</asp:ScriptManager>
```

这样，在客户端脚本中就可以直接调用 Web 服务的静态方法了。

作为 ASP.NET Web 服务(Web services)而创建的 AJAX 脚本服务(AJAX script services)，必须用 JavaScript 代理类(JavaScript proxy classes)来调用。调用该 AJAX 脚本服务的 JavaScript 代理类，是由引用该 AJAX 脚本服务的客户端页面控件 ScriptManager 在页面下载后，由运行库自动生成并链接到客户端页面的。

JavaScript 代理类具有与脚本服务相同的名称和许多附加属性。它的特征是有一组相同的方法，只不过采用了稍有扩展的签名，没有必要探究代理类的源代码。不过，如果想看一下它的结构，可从浏览器的地址栏中尝试调用 URL：http://.../service.asmx/js。

浏览器将下载一个 JavaScript 文件，将其保存到本地磁盘，用“记事本”打开来详阅。

JavaScript 代理类继承 Sys.Net.WebServiceProxy 基类，它提供了进行 JSON 调用的基本功能。

表 10-2　JavaScript 代理类的属性

属　　性	说　　明
Path	指出底层 Web 服务的 URL
Timeout	指出在调用超时之前允许方法运行的毫秒数
defaultSucceededCallback	指出调用成功时所调用的默认 JavaScript 回调函数
defaultFailedCallback	指出调用失败或超时情况下所调用的默认 JavaScript 回调函数(如果有)
defaultUserContext	指出要传递给成功回调或失败回调的默认 JavaScript 对象(如果有)

JavaScript 代理类的特征是具有表 10-2 中列出的属性。除了常规的一组参数之外，每个镜像的方法还有 3 个参数。这 3 个参数分别是：方法成功时所要调用的回调函数、方法失败或超时情况下所要调用的回调函数、传递给两个回调的上下文对象。

(3) 下面是从 ASP.NET AJAX 页面调用一个远程 AJAX 服务的一些示例代码：

```
function onclickgetServerTime()
{
  //window .alert ("calling onclickgetServerTime() ......");
  AjaxWebServices.TimeWebService.GetServerTime(onMethodComplete);
}
function onMethodComplete(results)
{   $get("Label1").innerHTML = results; }
```

在方法调用 GetServerTime(onMethodComplete)的实参中(无论结果如何)调用的成功回调函数 onMethodComplet 具有原型：function method(results, context, methodName)。

context 参数代表调用时指定的上下文对象。methodName 参数是一个设为服务方法名称的字符串。最后，对于成功调用时所调用的回调，results 参数是包含 JavaScript 版本的方法返回值的对象。对于失败的回调，此参数则代表 Sys.Net.WebServiceError 对象。

AJAX 完全是关于最广泛意义上的用户体验——连贯的感受、无闪烁更新、界面设计、资源聚合、实时数据等等。但只能利用浏览器和它的一套可编程性功能，主要是浏览器的对象模型、DOM 实现、对 DHTML 扩展的支持、CSS、JavaScript 和插件。

JavaScript 是构建和操纵 UI 的主要工具。用户界面任务的典型模式要求客户端使用 JavaScript 调用远程服务、接收 JSON 数据或 XML 数据，然后重新整理页面以显示更改。

纯 AJAX 体系结构是基于客户端与服务器的松散耦合，即实质上彼此独立的两个世界通过 HTTP 线路就 JSON 交换消息连接了起来。在纯 AJAX 体系结构中有一个基于 Web 服务的后端和一个 JavaScript 驱动的前端。构建有效的 HTML UI 完全取决于设计者或设计者选择的控件库。不过，这种分离机制使 Web 开发人员能够继续采用 Silverlight 等新兴技术创建交互性更强的用户界面，而不受服务器平台的限制。

实验 10　Web 服务

1. 实验目的

熟悉支持 AJAX 的 Web 服务的创建和访问技术，掌握用 C# 语言编写 Web 服务和用 Javascript 客户端脚本调用 Web 服务的方法。

2. 实验内容和要求

(1) 新建名字为 AjaxEnabledAddWebService 的网站。这里的正确路径为 http://localhost/AjaxEnabledAddWebService。

(2) 在【解决方案资源管理器】中，右击网站的名称 http://localhost/HelloWorldService，然后单击【添加新项】，新建 Web 服务 AddWebService.asmx，添加代码如下：

```
[System.Web.Script.Services.ScriptService]
public class AddWebService : System.Web.Services.WebService
{
    [WebMethod]
    public string Add(int a, int b)
    {
        int addition = a + b;
        string result = String.Format("The addition result is {0}.", addition.ToString());
        return result;
    }
}
```

(3) 新建调用 Add 方法的客户端 JavaScript 脚本文件 CallAddWebServiceMethod.js。

(4) 添加一个网页，通过客户端脚本文件调用 Web 服务。详细代码参见实验源码。

(5) 登录 http://www.onhap.com/，选自己喜欢的 Web 服务，进行编程练习。

(6) 登录 http://www.Webxml.com.cn/zh_cn/index.aspx，选自己喜欢的 Web 服务，进行编程练习。

第11章　在ASP.NET中使用XML

XML 被称为可扩展标记语言(eXtensible Markup Language)，它提供了一种保存数据的格式，数据可以通过这种格式很容易地在不同的应用程序之间实现共享。

XML 是专为 Web 设计的，.NET 也把 XML 作为应用程序之间传递数据的一种主要方法。本章将介绍 XML 的基本概念和如何访问 XML。

本章的学习目标

- 了解 XML 的基本概念
- 掌握如何利用 ADO.NET 访问 XML
- 掌握如何利用 XML 类访问 XML

11.1　XML 的基本概念

XML 是一种可以用来创建自己的标记的标记语言。它由万维网协会(W3C)创建，用来克服 HTML(即超文本标记语言(Hypertext Markup Language)，它是所有网页的基础)的局限。和 HTML 一样，XML 基于 SGML——标准通用标记语言(Standard Generalized Markup Language)。XML 是 SGML 上的一个子集，XML 包含了 SGML 很多特性，但是要比 SGML 简单得多。

XML 是一种类似于 HTML 的标记语言，但是 XML 不是 HTML 的替代品，XML 和 HTML 是两种不同用途的语言，其中最主要的区别是：XML 是专门用来描述文本的结构，而不是用于描述如何显示文本的，而 HTML 则是用来描述如何显示文本的。

XML 不像 HTML 那样提供了一组事先已经定义好的标记，而是提供了一个标准。利用这个标准，可以根据需要定义自己的新的标记。准确地说，XML 是一个元标记语言，它允许开发人员根据规则，制定各种各样的标记语言。

XML 是用来存放数据的，换句话来说，它可以作为微型数据库，这是最常见的数据型应用之一。可以利用相关的 XML API(MSXML DOM、JAVA DOM 等)对 XML 进行存取和查询。

总之，XML 是一种抽象的语言，它不如传统的程序语言那么具体。要深入地认识它，应该先从它的应用入手，选择一种需要的用途，然后再学习如何使用。

11.1.1　XML 的基本结构

下面就创建一个 XML 文档，用于保存学生信息。

【例 11-1】创建用于存放学生信息 XML 文档。

(1) 新建网站名为 XMLDemo，在【解决方案资源管理器】中，右击网站名，选择【添加新项】命令，在弹出的对话框中选择【XML 文件】模板，更改名称为 studentinfo.xml，创建 XML 文档。

(2) 在 studentinfo.xml 中输入如下数据。

```
<?xml version="1.0" standalone="yes" ?>
<!--下面存放的是学生信息-->
<students>
    <student>
      <no>11</no>
      <name>李明</name>
      <sex>男</sex>
      <birth>1988-10-01</birth>
      <address telephone="010-30008699">北京</address>
    </student>
    <student>
      <no>12</no>
      <name>陈倩</name>
      <sex>女</sex>
      <birth>1989-01-12</birth>
      <address telephone="021-30008655">上海</address>
    </student>
    …
</students>
```

第一行是 XML 版本说明，其作用是告诉浏览器或者其他处理程序：这个文档是 XML 文档。其中：version 表示文档遵守的 XML 规范的版本，如该例中的 version 是 1.0；standalone 表示文档内部包含文档类型定义 DTD。

注意：
版本说明必须是文档的第一行。

第二行是 XML 注释，其语法与 HTML 的注释一样。这并不是一种巧合，因为 XML 和 HTML 都是从 SGML 派生出来的。XML 文档注释不是强制性的，可以任意删除。

从第三行<students>标记一直到最后</students>标记，表示 XML 所包含的数据内容。

可以看出，XML 文档如同它的名字所指明的含义一样，是一个用标记语言创立的文档。它使用了自定义的各种标记来表示数据的含义。

11.1.2　标记、元素以及元素属性

标记是左尖括号(<)和右尖括号(>)之间的文本。有开始标记(例如 <name>)和结束标记(例如 </name>)。

元素是开始标记、结束标记以及位于二者之间的所有内容。在【例 11-1】中，<student>元素包含 5 个子元素：<no>、<name>、< sex>、<birth >和<address>。

属性是一个元素的开始标记中的名称-值对。在上面的示例中，telephone 是<address>元素的属性。

为了使一个 XML 文档结构完整，XML 必须遵守一定规则。常见的 XML 文档规则如下：

(1) 文档必须以 XML 版本声明开始。

(2) 含有数据的元素必须有起始标记和结束标记。每个起始标记必须以相应的结束标记结束。如果一个文档未能结束一个标记，浏览器将报告一个错误信息，并且不会以任何形式显示任何文档的内容。

(3) 不含数据并且仅使用一个标记的元素必须以/>结束。

(4) 文档只能包含一个能够包含全部其他元素的根元素。如：<students>元素。

(5) 元素只能嵌套不能重叠。

(6) 属性值必须加引号。如：<address telephone="021-30008655">中 telephone 属性的值不能写成< address telephone="021-30008655">。

11.1.3　XML 数据的显示

单独用 XML 是不能像页面那样显示数据的，必须使用某种格式化技术，比如 CSS 或者 XSL，才能显示 XML 标记创建的文档。

前面已经讲到 XML 是将数据和格式分离的。XML 文档本身不知道如何来显示数据，必须有辅助文件来帮助实现。因为 XML 取消了所有标识，包括 font，color 和 p 等风格样式的定义标识，因此 XML 全部是采用类似 DHTML 中 CSS 的方法来定义文档样式。

XML 中用来设定显示风格样式的文件类型通常有 XSL 和 CSS 两种。

1. 使用 XSL 显示 XML

XSL 全称为 eXtensible Stylesheet Language(可扩展样式表语言)，是用来设计 XML 文档显示样式的主要文件类型。它本身也是基于 XML 语言的。XSL 可以灵活地设置文档显示样式，XSL 可以将 XML 转化为其他的文档，比如 HTML 文档，这样，就可以在浏览器上显示了。

下面用一个简单的 XSL 样式表来说明如何显示 XML 文档。限于篇幅，本书无法详述 XSL，有兴趣的读者可以参考其他书籍。

【例 11-2】用 XSL 样式表显示 XML 文档。

(1) 打开网站 XMLDemo，在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【添加新项】命令，在弹出的对话框中选择【XSLT 文件】模板，新建 XSL 样式表，默认名称为 XSLTFile.xslt。

(2) 在 XSLTFile.xslt 文档中添加下面代码。

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">
    <xsl:template match="/">
      <html>
        <body>
          <h2>学生信息</h2>
          <table border="1">
            <tr bgcolor="#9acd32">
              <th align="left">学号</th>
              <th align="left">姓名</th>
              <th align="left">性别</th>
              <th align="left">年龄</th>
              <th align="left">住址</th>
            </tr>
            <xsl:for-each select="students/student">
              <tr>
                <td>   <xsl:value-of select="no" />   </td>
                <td>   <xsl:value-of select="name" /> </td>
                <td>   <xsl:value-of select="sex" /> </td>
                <td>   <xsl:value-of select="birth" />   </td>
                <td>   <xsl:value-of select="address" />   </td>
              </tr>
            </xsl:for-each>
          </table>
        </body>
      </html>
    </xsl:template>
</xsl:stylesheet>
```

前 3 行代码是系统自动生成的，包括版本说明和命名空间引用等。<template>元素定义了一个 HTML 表格。表格的单元格将用于 XML 文档数据显示。

(3) 打开 Default.aspx 网页，在页面上添加一个 XML 控件。该控件在工具箱的【标准】选项卡中。

(4) 设置 XML 控件的 DocumentSource 和 TransformSource 属性，将 DocumentSource 指向 studentinfo.xml 文档，TransformSource 指向 XSLTFile.xslt 文档。结果如下：

```
<asp:Xml ID="Xml1" runat="server" DocumentSource="~/studentinfo.xml"
            TransformSource="~/XSLTFile.xslt">   </asp:Xml>
```

(5) 保存并运行网站，studentinfo.xml 中内容就以表格的形式呈现在浏览器上。如图 11-1 所示。

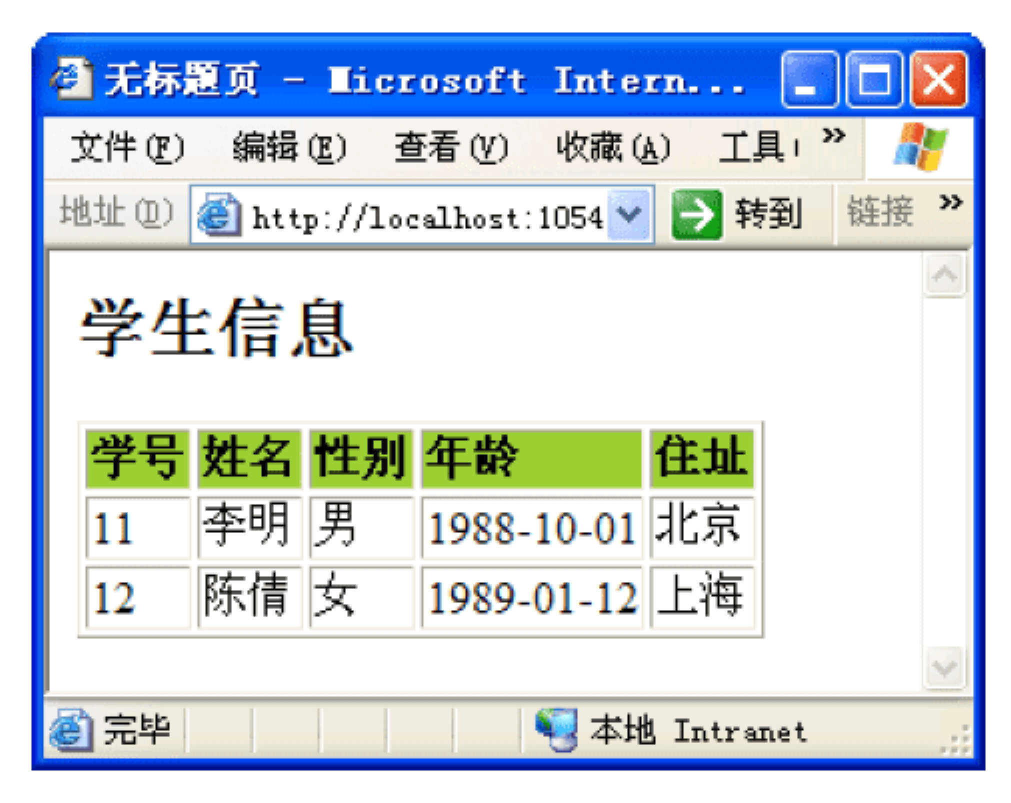


图 11-1　studentinfo.xml 的显示效果

2. 使用 CSS 显示 XML

CSS 大家很熟悉了，全称是 Cascading Style Sheets(层叠样式表)，是目前用来在浏览器上显示 XML 文档的主要方法。

下面用一个简单的 CSS 样式表来说明如何显示 XML 文档。

【例 11-3】用 CSS 样式表显示 XML 文档。

(1) 打开网站 XMLDemo，在【解决方案资源管理器】中，右击网站名，选择【添加新项】命令，在弹出的对话框中选择【样式表】模板，新建 CSS 样式表，默认名称为 StyleSheet.css。

(2) 在 StyleSheet.css 样式表中添加如下所示的代码。

```
name
{
    font-family:  宋体, Arial, Helvetica, sans-serif;
    font-size: large;
}
no,sex, birth, address
{
    font-family:  宋体, Arial, Helvetica, sans-serif;
    font-size: small;
}
```

(3) 打开 studentinfo.xml 文档，添加下面一行代码，建立与 StyleSheet.css 样式表的关联。

```
…
<?xml-stylesheet href="StyleSheet.css" type="text/css" ?>
<students>
…
```

(4) 在浏览器上打开 studentinfo.xml。效果如图 11-2 所示。



图 11-2　studentinfo.xml 的显示效果

11.2　使用 ADO.NET 访问 XML

在前面已经讨论了如何使用 ADONET 访问数据库的问题。数据库是进行数据存储和管理的一种习惯的方式，现在，XML 已逐步成为数据存储的一种新的方式，因此可以考虑将数据保存在 XML 文档中，并采用一定的方法对它进行管理。ADONET 提供了对 XML 数据访问的功能。

下面介绍如何使用 ADONET 访问 XML 数据。

11.2.1　将数据库数据转换成 XML

为了将数据库数据转换成 XML 文档，需要使用 DataSet 的 WriteXml 方法。WriteXml 方法只要指明要保存的 XML 文档的路径和文件名，就可以将 DataSet 中数据以 XML 的形式保存到 XML 文档中。

【例 11-4】将数据库数据转换成 XML 文档。

(1) 打开网站 XMLDemo，在【解决方案资源管理器】中，右击网站名，选择【添加新项】命令，在弹出的对话框中选择【SQL Server 数据库】模板，更改名称为 MyDatabase.mdf，创建数据库。

(2) 在 MyDatabase.mdf 数据库中创建 student 表，并输入模拟数据。表的关系模式如下：

Student(no，name，sex，birth，address)

(3) 在 web.config 配置文件中，修改<connectionStrings/>标记如下：

```
 <connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=.\SQLEXPRESS;
       AttachDbFilename=|DataDirectory|\ MyDatabase.mdf;Integrated Security=True;User Instance=True"/>
</connectionStrings>
```

(4) 新建网页 write_xml.aspx，在页面上添加一个 Label 控件。

(5) 给 write_xml.aspx 添加后台代码如下。

```
//引用命名空间
using System.Data.SqlClient;
…
protected void Page_Load(object sender, EventArgs e)
{
    //从 Web.Config 中取出数据库连接串
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    //创建连接对象
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //创建 DataSet 对象
    DataSet ds = new DataSet();
    //打开连接
    sqlconn.Open();
    //创建适配器对象
    SqlDataAdapter sqld = new SqlDataAdapter("select no,name,sex, CONVERT(char(10), birth,20) AS
birth,address from student", sqlconn);
    //利用适配器方法添加数据给 DataSet
    sqld.Fill(ds, "student");
    //将 DataSet 数据写成 XML 文本
    ds.WriteXml(Server.MapPath("students.xml"));
    sqlconn.Close();
    Label1.Text = "写入成功";
 }
```

(6) 运行程序，打开程序运行时保存的 students.xml 文件，内容如下：

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <student>
    <no>1</no>
    <name>张小兵</name>
    <sex>男</sex>
    <birth>1989-06-28</birth>
    <address>上海</address>
  </student>
  <student>
    <no>2</no>
    <name>李明</name>
    <sex>男</sex>
    <birth>1988-05-01</birth >
    <address>上海</address>
  </student>
  …
</NewDataSet>
```

可以看出，这个文档保存了 student 数据表中所有的数据。其中使用<NewDataSet>作为根结点标记，<student>作为每个记录的标记(student 是 sqld.Fill(ds, "student")语句中使用的名字)，另外，每个字段的名字作为数据元素的标记名。

11.2.2　读取 XML 文档

使用 DataSet 的 ReadXml 方法可以读取所有 XML 文档数据。下面通过【例 11-5】来说明读取 XML 文档的方法。

【例 11-5】读取 XML 文档。

(1) 打开网站 XMLDemo。新建网页 read_xml.aspx，在页面上添加 1 个 GridView 控件。

(2) 给 read _xml.aspx 添加后台代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
        DataSet ds = new DataSet();
        //读取 XML 文本数据到 DataSet 数据集
        ds.ReadXml(Server.MapPath("students.xml"));
        //绑定数据源
        GridView1.DataSource = ds.Tables[0].DefaultView;
        GridView1.DataBind();
}
```

(3) 程序执行结果如图 11-3 所示。



no	name	sex	birth	address
1	张小兵	男	1989-06-28	上海
2	李明	男	1988-05-01	上海
3	王彬	男	1989-09-25	郑州
4	陈飞	男	1988-06-08	洛阳
5	王燕	女	1989-12-11	沈阳
6	陈武	男	1987-07-21	广州
7	刘华	女	1990-03-15	广州
8	翟力	女	1989-06-11	深圳
9	大侠	男	1989-06-01	沈阳

图 11-3　read_xml.aspx 的运行效果

11.2.3　编辑 XML 文档

编辑 XML 文档的方法也很简单，只要使用 DataSet 的 ReadXml 方法把 XML 数据读到 DataSet 中之后，修改相应的记录值，再使用 DataSet 的 WriteXml 方法保存 XML 文档就可以了。

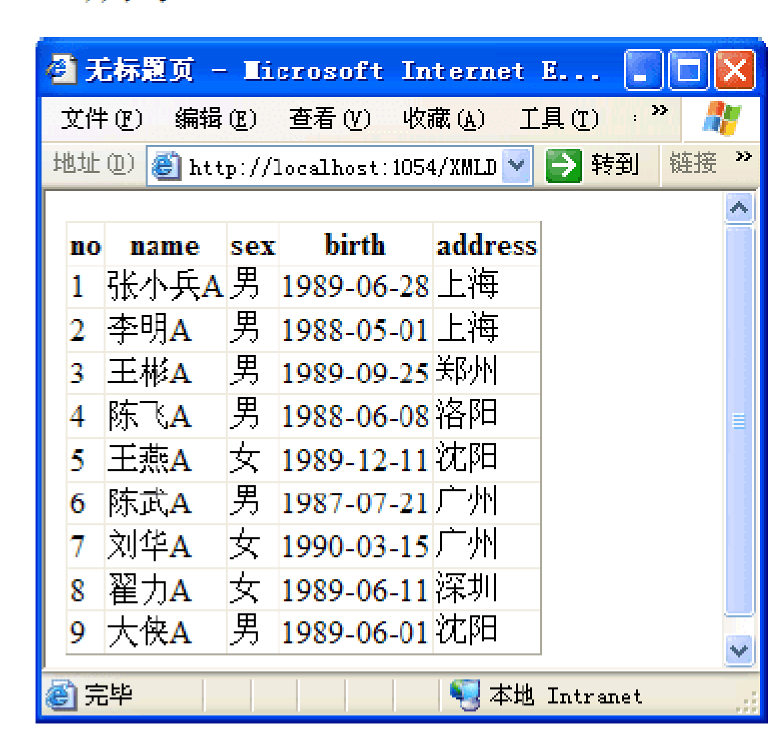
【例 11-6】编辑 XML 文档。

(1) 打开网站 XMLDemo。新建网页 edit_xml.aspx，在页面上添加 1 个 GridView 控件。

(2) 给 edit _xml.aspx 添加后台代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    //建立 DataSet 对象
    DataSet ds = new DataSet();
    ds.ReadXml(Server.MapPath("students.xml"));
    //建立 DataTable 对象
    DataTable dtable;
    //建立 DataRowCollection 对象
    DataRowCollection coldrow;
    //建立 DataRow 对象
    DataRow drow;
    //将数据表 tabstudent 的数据复制到 DataTable 对象
    dtable = ds.Tables[0];
    //用 DataRowCollection 对象获取这个数据表的所有数据行
    coldrow = dtable.Rows;
    //修改操作，逐行遍历，取出各行的数据
    for (int inti = 0; inti < coldrow.Count; inti++)
    {
        drow = coldrow[inti];
        //给每位学生姓名后加上字母 A
        drow[1] = drow[1] + "A";
    }
    //将 DataSet 数据写成 XML 文本
    ds.WriteXml(Server.MapPath("students.xml"));
    //绑定数据源
    GridView1.DataSource = ds.Tables[0].DefaultView;
    GridView1.DataBind();
}
```

(3) 程序执行结果如图 11-4 所示。



no	name	sex	birth	address
1	张小兵A	男	1989-06-28	上海
2	李明A	男	1988-05-01	上海
3	王彬A	男	1989-09-25	郑州
4	陈飞A	男	1988-06-08	洛阳
5	王燕A	女	1989-12-11	沈阳
6	陈武A	男	1987-07-21	广州
7	刘华A	女	1990-03-15	广州
8	翟力A	女	1989-06-11	深圳
9	大侠A	男	1989-06-01	沈阳

图 11-4　edit _xml.aspx 的运行效果

经过修改后，每个学生姓名增加了大写字母 A。可以看到，使用 ADO.NET 编辑 XML 文档，实际上就是对 DataSet 数据集数据的编辑，使用非常简单。

11.2.4　将 XML 写入数据库

XML 文本内容写入数据库与数据库的数据转换为 XML 是个相反的过程，需要用到 DataAdapter 的 Update 方法。下面列举一个简单的例子，将 studentinfo.xml 文档中仅有的学号为 11 和 12 的两位学生的数据，写入到数据库的 student 表中。

【例 11-7】将 XML 写入数据库。

(1) 打开网站 XMLDemo。新建网页 xml_table.aspx，在页面上添加 1 个 GridView 控件。

(2) 给 xml_table .aspx 添加后台代码如下。

```
//引用命名空间
using System.Data.SqlClient;
…
protected void Page_Load(object sender, EventArgs e)
{
    //从 Web.Config 中取出数据库连接串
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    //创建连接对象
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //创建 DataSet 对象
    DataSet ds = new DataSet();
    //打开连接
    sqlconn.Open();
    //创建适配器对象
    SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
    //利用适配器方法添加数据给 DataSet
    sqld.Fill(ds, "student");
    DataTable dt = ds.Tables["student"];
    //读取 XML
    dt.ReadXml(Server.MapPath("studentinfo.xml"));
    //自动生成提交语句
    SqlCommandBuilder objcb = new SqlCommandBuilder(sqld);
    //提交数据库
    sqld.Update(ds, "student");
    //绑定数据源
    GridView1.DataSource = ds.Tables["student"].DefaultView;
    GridView1.DataBind();
}
```

(3) 程序执行结果如图 11-5 所示。可以看到，新增了 11 和 12 号两位学生。

no	name	sex	birth	address
1	张小兵	男	1989-6-28 0:00:00	上海
2	李明	男	1988-5-1 0:00:00	上海
3	王彬	男	1989-9-25 0:00:00	郑州
4	陈飞	男	1988-6-8 0:00:00	洛阳
5	王燕	女	1989-12-11 0:00:00	沈阳
6	陈武	男	1987-7-21 0:00:00	广州
7	刘华	女	1990-3-15 0:00:00	广州
8	翟力	女	1989-6-11 0:00:00	深圳
9	大侠	男	1989-6-1 0:00:00	沈阳
11	李明	男	1988-10-1 0:00:00	北京
12	陈倩	女	1989-1-12 0:00:00	上海

图 11-5　xml_table.aspx 的运行效果

11.2.5　将 XML 数据转换为字符串

前面讲到的所有方法都是使用 DataSet 来进行数据处理的。在实际工作中，如果希望进行 XML 数据传输，那么把 XML 数据读出之后形成字符串，即把数据当成字符串法进行处理。例如可以把数据写在一个普通的 E-mail 中发送给其他的用户，对方就可以采用普通字符串处理的方法得到数据。为了能够完成上述功能，DataSet 还提供了将 XML 数据转换为字符串的方法 GetXml。

【例 11-8】编辑 XML 文档。

(1) 打开网站 XMLDemo。新建网页 Trans_xml.aspx，在页面上添加 1 个 Label 控件。

(2) 给 Trans_xml.aspx 添加后台代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    DataSet ds = new DataSet();
    //读取 XML 文本数据到 DataSet 数据集
    ds.ReadXml(Server.MapPath("students.xml"));
    //将 DataSet 的数据转换为字符串
    Label1.Text = ds.GetXml();
}
```

(3) 程序执行结果如图 11-6 所示。

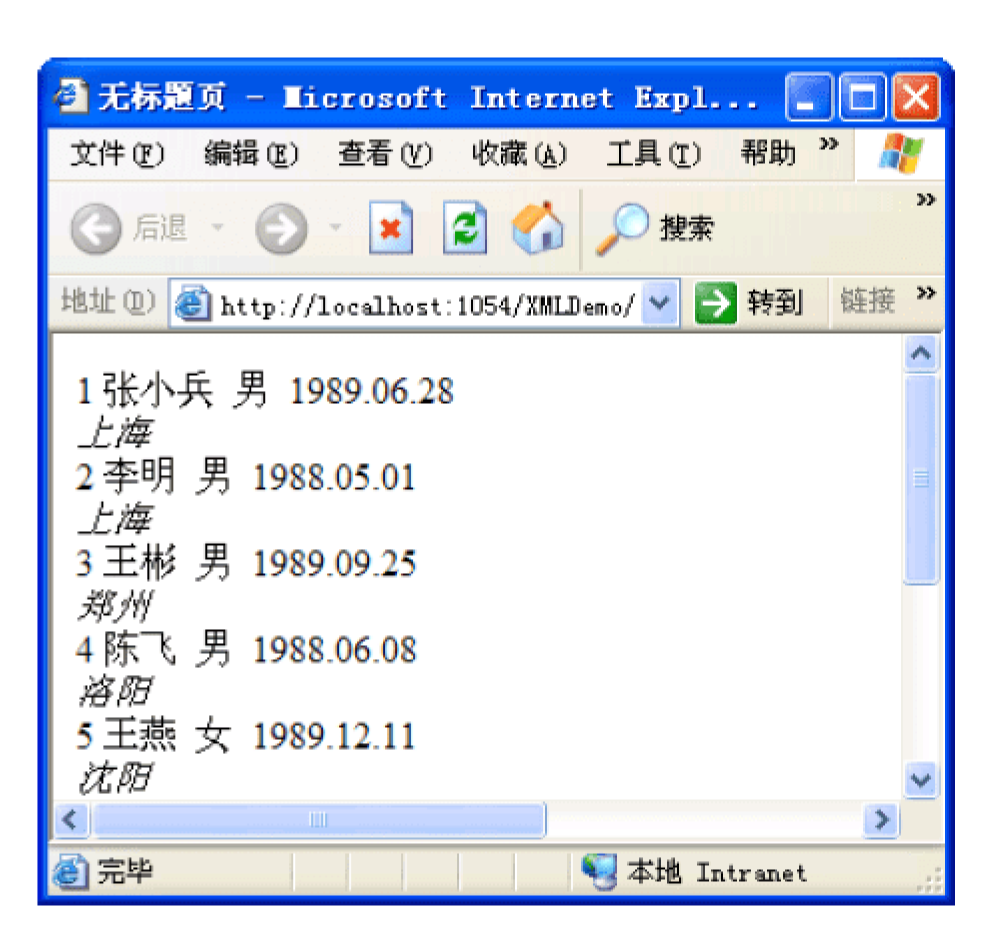


图 11-6　Trans_xml.aspx 的运行效果

11.3　使用.NET 的 XML 类访问 XML

ASP.NET 通过 System.Xml 命名空间为开发人员提供了操作 XML 的所有功能。该命名空间包括许多类，常用类如表 11-1 所示。

表 11-1　System.Xml 命名空间常用类

类	说　明
XmlReader	抽象的读取器类，提供快速、没有缓存的 XML 数据。XmlReader 是只向前的，类似于 SAX 分析器
XmlWriter	抽象类的写入器，以流或文件的格式提供快速、没有缓存的 XML 数据
XmlTextReader	扩展 XmlReader，提供访问 XML 数据的快速只向前流
XmlTextWriter	扩展 XmlWriter，快速生成只向前的 XML 流
XmlNode	抽象类，表示 XML 文档中一个节点的类。XML 命名空间中几个类的基类
XmlDocument	扩展 XmlNode，W3C DOM 的实现，给出 XML 文档在内存中的树形表示，可以浏览和编辑它们
XmlDataDocument	扩展 XmlDocument，即从 XML 数据中加载的文档，或从 ADO.NET DataSet 的关系数据中加载的文档，允许把 XML 和关系数据混合在同一个视图中
XmlResolver	抽象类，分析基于 XML 的外部资源，例如 DTD 和模式引用，也可以用于处理和元素
XmlUrlResolver	扩展 XmlResolver，用 URI(Uniform Resource Identifier)解析外部资源

有关 XML 类编程技术内容较多，本书在此仅通过下面的【例 11-9】和【例 11-10】来简单说明 XML 类编程的过程。

【例 11-9】将数据库 student 表中男同学的数据保存到 XML 文档中。

(1) 打开网站 XMLDemo。新建网页 write_xml2.aspx，在页面上添加 1 个 Label 控件。

(2) 给 write_xml2.aspx 添加后台代码如下。

```
//引用命名空间
using System.Data.SqlClient;
using System.Xml;
…
protected void Page_Load(object sender, EventArgs e)
{
    //从 Web.Config 中取出数据库连接串
    string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    //创建连接对象
    SqlConnection sqlconn = new SqlConnection(sqlconnstr);
    //创建 DataSet 对象
    DataSet ds = new DataSet();
    //打开连接
    sqlconn.Open();
    //创建适配器对象
    SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
    //利用适配器方法添加数据给 DataSet
    sqld.Fill(ds, "student");
    DataTable dt = ds.Tables["student"];
    //创建 XML 文档
    XmlDocument xmldoc = new XmlDocument();
    //写入版本说明
    XmlDeclaration xmldecl = xmldoc.CreateXmlDeclaration("1.0", "utf-8", "yes");
    xmldoc.AppendChild(xmldecl);
    //写入根结点
    XmlElement students = xmldoc.CreateElement("students");
    xmldoc.AppendChild(students);
    //对每一行循环
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        if (dt.Rows[i][2].ToString()=="男")
        {
            XmlElement student = xmldoc.CreateElement("student");
            //循环每一列
            for (int j = 0; j < dt.Columns.Count; j++)
            {
                XmlElement colname = xmldoc.CreateElement(dt.Columns[j].ColumnName);
                colname.InnerText = dt.Rows[i][j].ToString();
                student.AppendChild(colname);
            }
            students.AppendChild(student);
        }
```

```
        }
        //将 xmldoc 数据写成 XML 文本
        xmldoc.Save(Server.MapPath("student_men.xml"));
        sqlconn.Close();
        Label1.Text = "写入成功";
    }
```

(3) 运行程序，打开程序运行时保存的 student_men.xml 文件，可以看出 XML 文档中只保存了男生的数据。

【例 11-10】将 studentinfo.xml 文档中男生数据写到 student 数据表中。

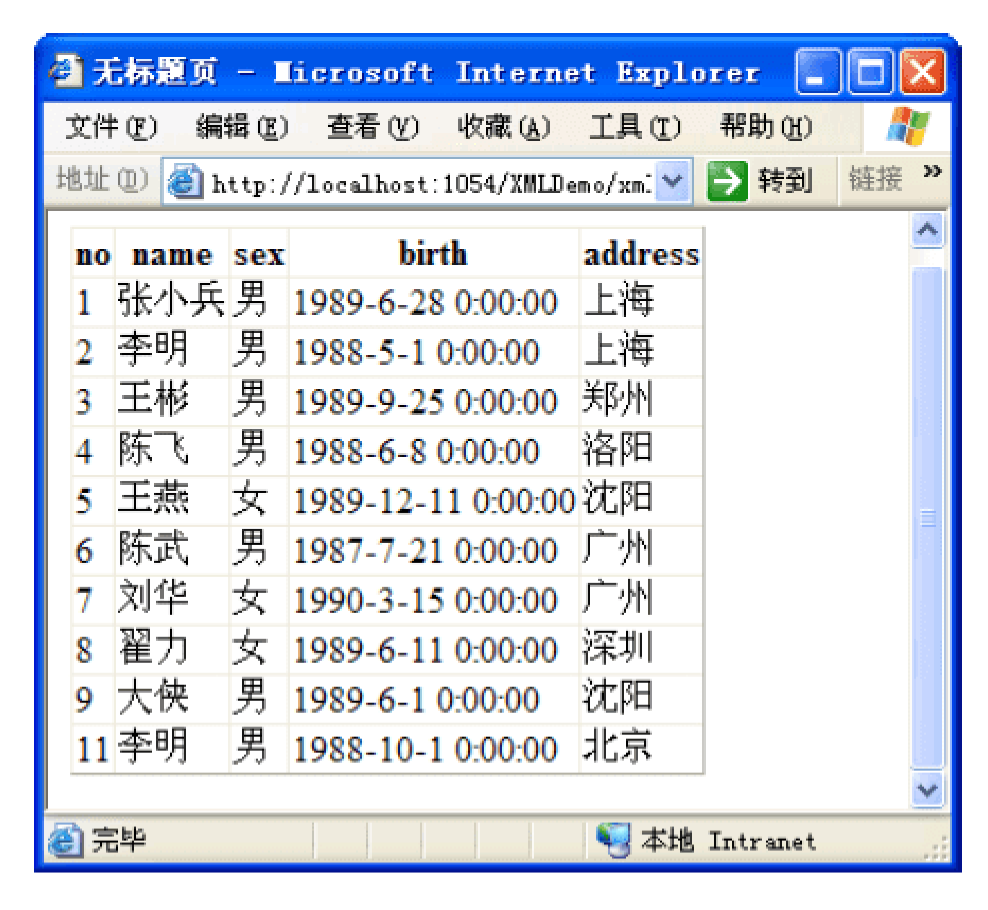
(1) 打开网站 XMLDemo。新建网页 xml_table2.aspx，在页面上添加 1 个 GridView 控件。

(2) 给 xml_table2 .aspx 添加后台代码如下。

```
    //引用命名空间
    using System.Data.SqlClient;
    using System.Xml;
    …
    protected void Page_Load(object sender, EventArgs e)
    {
        //从 Web.Config 中取出数据库连接串
        string sqlconnstr = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
        //创建连接对象
        SqlConnection sqlconn = new SqlConnection(sqlconnstr);
        //创建 DataSet 对象
        DataSet ds = new DataSet();
        //打开连接
        sqlconn.Open();
        //创建适配器对象
        SqlDataAdapter sqld = new SqlDataAdapter("select * from student", sqlconn);
        //利用适配器方法添加数据给 DataSet
        sqld.Fill(ds, "student");
        DataTable dt = ds.Tables["student"];
        //创建 XML 文档
        XmlDocument xmldoc = new XmlDocument();
        //读取 xml
        xmldoc.Load(Server.MapPath("studentinfo.xml"));
        //得到整个元素
        XmlElement xmle = xmldoc.DocumentElement;
        //遍历每个元素
        for (int i = 0; i < xmle.ChildNodes.Count; i++)
        {
            XmlNode xmln = xmle.ChildNodes[i];
            if (xmln.ChildNodes[2].InnerText == "男")
            {
```

```
                DataRow newrow = dt.NewRow();
                newrow["no"] = xmln.ChildNodes[0].InnerText;
                newrow["name"] = xmln.ChildNodes[1].InnerText;
                newrow["sex"] = xmln.ChildNodes[2].InnerText;
                newrow["birth"] = xmln.ChildNodes[3].InnerText;
                newrow["address"] = xmln.ChildNodes[4].InnerText;
                dt.Rows.Add(newrow);
            }
        }
        //自动生成提交语句
        SqlCommandBuilder objcb = new SqlCommandBuilder(sqld);
        //提交数据库
        sqld.Update(ds, "student");
        //绑定数据
        GridView1.DataSource = ds.Tables["student"].DefaultView;
        GridView1.DataBind();
    }
```

(3) 程序执行结果如图 11-7 所示。可以看到 studentinfo.xml 文档中 11 号男生数据被写到了数据库的 student 表中。



no	name	sex	birth	address
1	张小兵	男	1989-6-28 0:00:00	上海
2	李明	男	1988-5-1 0:00:00	上海
3	王彬	男	1989-9-25 0:00:00	郑州
4	陈飞	男	1988-6-8 0:00:00	洛阳
5	王燕	女	1989-12-11 0:00:00	沈阳
6	陈武	男	1987-7-21 0:00:00	广州
7	刘华	女	1990-3-15 0:00:00	广州
8	翟力	女	1989-6-11 0:00:00	深圳
9	大侠	男	1989-6-1 0:00:00	沈阳
11	李明	男	1988-10-1 0:00:00	北京

图 11-7　xml_table2.aspx 的运行效果

11.4　XML 的应用和发展前景

虽然人们对 XML 的某些技术标准尚有争议，但是人们已经普遍认识到 XML 的作用和巨大潜力，下面对 XML 应用和发展前景作一个简要介绍。

11.4.1　XML 的应用

作为互联网的新技术，XML 的应用非常广泛，可以说 XML 已经渗透到了互联网的各个

角落。考察现在的 XML 应用，可以大致将它分为以下几种方面。

1. 数据交换

利用 XML 在应用程序之间作数据交换已不是什么秘密了，毫无疑问应被列为第一位。那么为什么 XML 在这个领域里的地位这么重要呢？原因就是 XML 使用元素和属性来描述数据。在数据传送过程中，XML 始终保留了诸如父、子关系这样的数据结构。几个应用程序可以共享和解析同一个 XML 文件，不必使用传统的字符串解析或拆解过程。

相反，普通文件不对每个数据段做描述，也不保留数据关系结构。使用 XML 做数据交换可以使应用程序更具有弹性，因为可以用位置(与普通文件一样)或用元素(从数据库)来存取 XML 数据。

2. Web 服务

Web 服务是最令人激动的革命之一，它让使用不同系统和不同编程语言的人们能够相互交流和分享数据。其基础在于 Web 服务器用 XML 在系统之间交换数据。交换数据通常用 XML 标记，能使协议取得规范一致，比如在简单对象处理协议(Simple Object Access Protocol，SOAP)平台上。

SOAP 可以在用不同编程语言构造的对象之间传递消息。这意味着一个 C#对象能够与一个 Java 对象进行通讯。这种通讯甚至可以发生在运行于不同操作系统上的对象之间。DCOM，CORBA 或 Java RMI 只能在紧密耦合的对象之间传递消息，SOAP 则可在松耦合对象之间传递消息。

3. 内容管理

XML 只用元素和属性来描述数据，而不提供数据的显示方法。这样，XML 就提供了一个优秀的方法来标记独立于平台和语言的内容。

使用像 XSLT 这样的语言能够轻易地将 XML 文件转换成各种格式文件，比如 HTML，WML，PDF，EDI 等等。XML 具有的能够运行于不同系统平台之间和转换成不同格式目标文件的能力使得它成为内容管理应用系统中的优秀选择。

4. Web 集成

现在有越来越多的设备支持 XML 了。使得 Web 开发商可以在个人电子助理和浏览器之间用 XML 来传递数据。

为什么将 XML 文本直接送进这样的设备去呢？这样做的目的是让用户更多地自己掌握数据显示方式，更能体验到实践的快乐。常规的客户/服务(C/S)方式为了获得数据排序或更换显示格式，必须向服务器发出申请；而 XML 则可以直接处理数据，不必经过向服务器申请查询，返回结果这样的双向“旅程”，同时设备也不需要配制数据库，甚至还可以对设备上的 XML 文件进行修改并将结果返回给服务器。

5. 配置文件

许多应用都将配置数据存储在各种文件里，比如.INI 文件。虽然这样的文件格式已经使

用多年并一直很好用，但是 XML 还是以更为优秀的方式为应用程序标记配制数据。使用.NET 里的类，如 XmlDocument 和 XmlTextReader，将配置数据标记为 XML 格式，能使其更具可读性，并能方便地集成到应用系统中去。使用 XML 配制文件的应用程序能够方便地处理所需数据，不用像其他应用那样要经过重新编译才能修改和维护应用系统。

11.4.2　XML 的发展前景

XML 自推出以来，尤其是在 1998 年 2 月成为 W3C 推荐标准以来，受到了广泛的支持。各大软件厂商如 IBM、Microsoft、Oracle、Sun 等都积极支持并参与 XML 的研究和产品化工作，先后推出了支持 XML 的产品或者改造原有的产品以支持 XML，W3C 也一直致力于完善 XML 的整个理论体系。

XML 虽然获得了极大的支持，但是它还有一段路要走。首先，XML 的规则还有许多技术细节没有解决。其次，现在虽然出现了一些 XML 工具和应用，但是其市场反应还有待进一步观察。另外如何让更多的人迅速学会使用 XML，并利用它进行开发，进而促进 XML 的应用也是一个问题。XML 的出现和迅猛发展并不意味着 HTML 即将退出互联网舞台。由于 HTML 的易学易用和非常多的工具支持，HTML 将在较长的时间里继续在 Web 舞台上充当主角。但如果用户想超越 HTML 的范围，XML 将是最佳的选择。

另外，由于 XML 是用于自定义标记的元语言，任何个人、公司和组织都可以利用它来定义自己的标记语言，这虽然是 XML 的魅力和灵活性之所在，但同时也是 XML 的最大问题之所在。如果每个人、公司和组织都定义了自己的标记语言，它们之间的通信就会出现困难。因此在一些领域先后出现了一些标准化组织，它们的任务就是规范本领域的标记语言，形成统一的标准，使得在本领域内的通信成为可能。但在标准推出并得到广泛认可之前，各自为政的局面将继续下去。更糟糕的是，由于对应用的理解不一致和商业利益等原因，同一个领域也许还有多个标准化组织，它们形成的标记语言并不完全兼容，使得采取不同标准的计算机仍然难以通信。

无论如何，XML 的出现使互联网跨入了一个新的阶段，它已成为因特网领域中一个重要的数据交换标准和开发平台。没有 XML 就没有 Web 服务，也就没有今天构建应用软件的轰轰烈烈的 SOA(Service Oriented Architecture)。XML 的诞生已经而且将继续促使全新种类的基础架构和应用程序的产生，而这些新的基础架构和应用程序又将需要新的软件和硬件工具。可以预测，无论是在软件还是硬件上，XML 都将开辟一系列的新市场，促成互联网上新的革命。

实验 11　XML 编程

1. 实验目的

进一步熟悉 XML 概念，掌握 ASP.NET 中 XML 编程方法。

2. 实验内容和要求

(1) 将 MyDatabase.mdf 数据库 student 表的数据写成 XML 文档，要求只转出 name、sex、birth 这 3 列，XML 文档的名称为 stu.xml。

(2) 创建 XSL 文件，并以表格的形式显示 stu.xml。

(3) 在 MyDatabase.mdf 中新建 men 数据表，将 stu.xml 文档中男生信息写到 men 表中。

(4) 将 stu.xml 文档中学生姓名 name 后面增加“学生”字样。

(5) 在 stu.xml 文档中新增一个学生的数据。

第12章　开 发 实 例

在网络经济与电子商务迅猛发展的今天，越来越多的企业认识到建立网站的必要性。网站是展示自己产品和提升企业形象的网络平台。但是如何有效地发布产品信息、服务信息和企业信息，在各种资源调配上做到管理有序，这都是对企业网络平台的重大挑战。

本章将介绍一个典型的企业网站。通过学习，读者将会对企业网站有一个系统认识。在此基础上，调研某一个企业的自身的需求，便可以制作实用的企业网站。

本章的学习目标

- 进一步熟悉 ASP.NET 编程技术
- 掌握 Web 控件使用方法
- 如何让 ADO.NET 编程更加简洁
- 熟悉网站的制作过程

12.1　系统设计

结合中小企业的实际。在需求分析的基础上，给出如下设计：概念结构设计、数据库设计和功能设计。

12.1.1　需求分析

企业网站的栏目和功能各不相同。通过对中小企业的调查分析，开发小组认为中小企业网站主要的栏目和功能应该包括：企业简介，让用户了解企业文化、理念、历史和规模；联系方式，让用户可以及时与企业沟通；企业新闻，让用户了解企业最新的活动、发展动态和优惠措施等；产品和服务，介绍产品的图片、规格、型号、价格、功能等信息，介绍企业所提供的各项服务；同时提供网站后台管理功能。

12.1.2　概念结构设计

系统的 E-R 图(图中省略了实体和联系的属性)，如图 12-1 所示，每个实体及属性如下。

新闻信息：流水号、新闻标题、新闻内容、新闻类别、添加时间、阅读次数。

新闻类别：流水号、新闻类别。

产品：流水号、产品名称、产品价格、产品图片、产品类别、产品介绍。

产品类别：流水号、产品类别。

用户：用户名、密码、真实姓名、电话、地址、邮编。

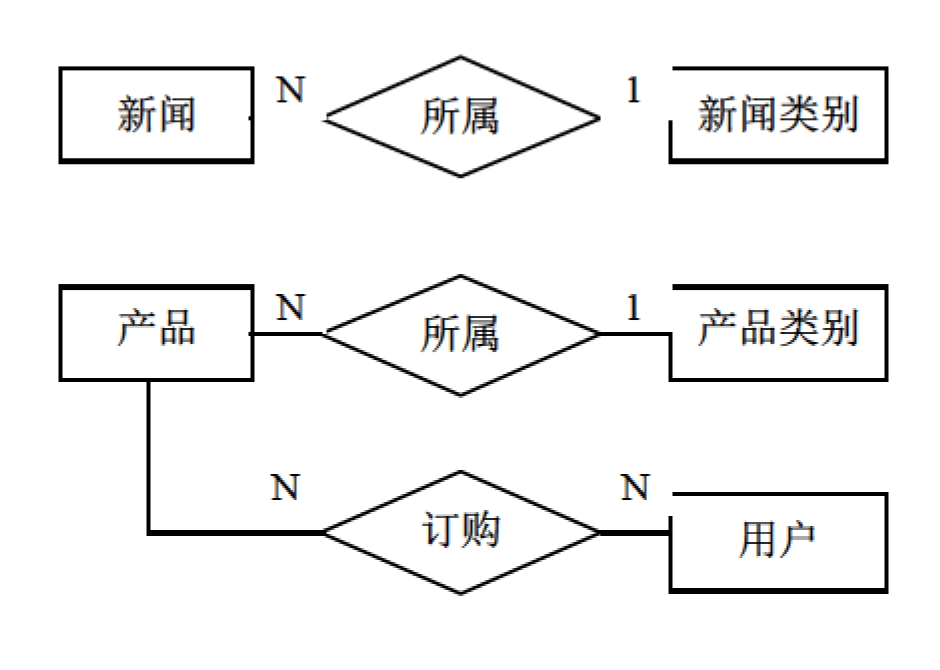


图 12-1　E-R 图

12.1.3　数据库设计

在图 12-1 所示的 E-R 图中，有 5 个实体、1 个多对多联系和 2 个一对多联系。由于每个实体可以用一张表表示，每个多对多联系也可以用一张表表示，而一对多的联系不需要建新表，所以，把 E-R 图转换成数据库的 6 张表就可以了。

这 6 张表分别是新闻信息表、新闻类别表、产品表、产品类别表、订单表和用户表。表的结构如表 12-1 至表 12-6 所示。

表 12-1　新闻信息表

列　　名	数 据 类 型	长　　度	说　　明
流水号	Bigint	8	主键
新闻标题	Nvarchar	50	
新闻内容	Ntext	16	
新闻类别	Nvarchar	10	外键
添加时间	smalldatetime	4	
阅读次数	Int	4	默认为 0

表 12-2　新闻类别表

列　　名	数 据 类 型	长　　度	说　　明
流水号	Bigint	8	主键
新闻类别	Nvarchar	50	

表 12-3　产品表

列　　名	数 据 类 型	长　　度	说　　明
流水号	Bigint	8	主键
产品名称	Nvarchar	50	
产品价格	Int	4	
产品图片	Varchar	50	图片文件名
产品类别	Varchar	10	外键
产品介绍	Ntext	16	

表 12-4　产品类别表

列　　名	数 据 类 型	长　　度	说　　明
流水号	Bigint	8	主键
产品类别	Nvarchar	10	

表 12-5　用户表

列　　名	数 据 类 型	长　　度	说　　明
用户名	Nvarchar	20	主键
密码	Nvarchar	10	
真实姓名	Nvarchar	50	
电话	Nvarchar	50	
地址	Nvarchar	50	
邮编	Nvarchar	6	
管理员标志	Bit	1	默认 0，表示一般用户

表 12-6　订单表

列　　名	数 据 类 型	长　　度	说　　明
流水号	Bigint	8	主键
产品流水号	Bigint	8	
订购数量	Int	4	
用户名	Nvarchar	20	
订购日期	Datetime	8	
处理标志	Bit	1	默认 0，表示未处理

12.1.4　功能设计

网站功能包括：产品、新闻、联系我们、产品订购、产品管理、产品添加、新闻管理、新闻添加、订单管理等，如图 12-2 所示。

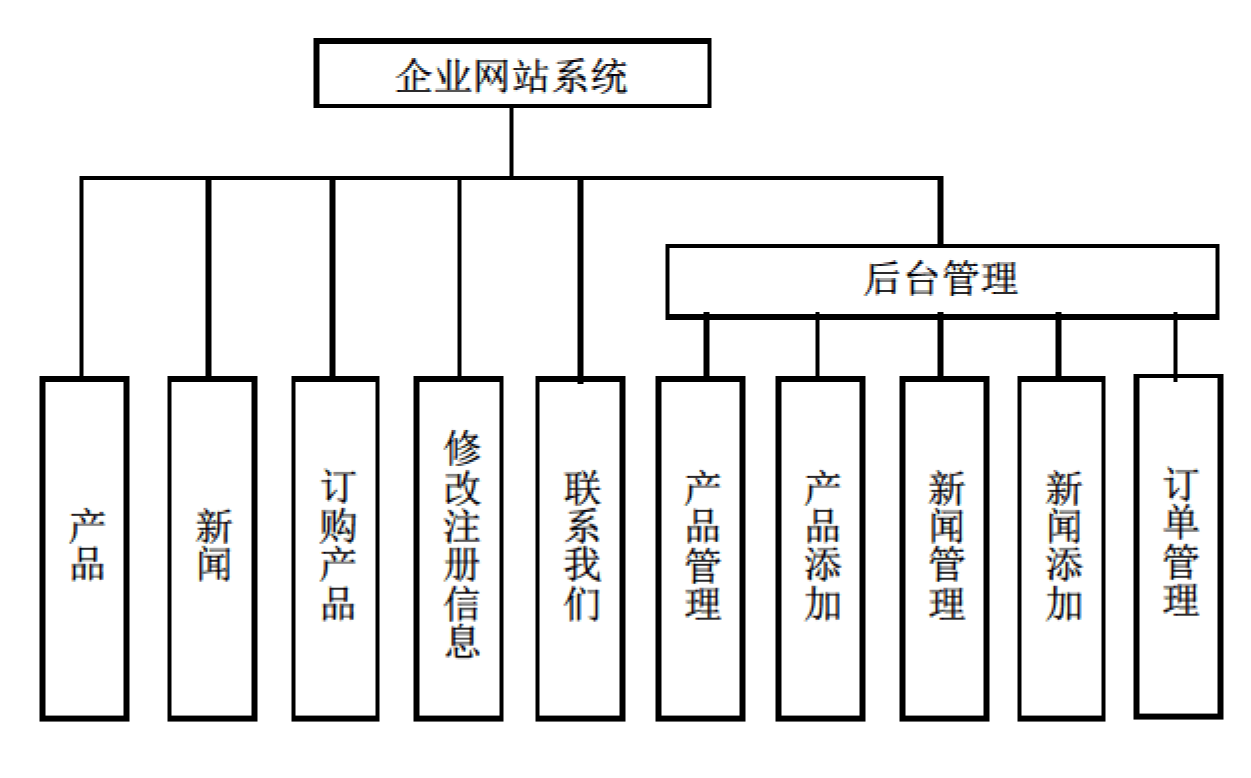


图 12-2　功能设计

12.2　程序设计

首先启动 VWD 2008，新建网站。然后在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【添加新项】命令，在弹出的对话框中选择【SQL Server 数据库】模板，创建名称为“实例数据库.mdf”的数据库。最后，在数据库中建立表 12-1 至表 12-6 所示的数据表。

下面详细介绍程序设计。

12.2.1　设置数据库连接信息

Web.Config 是 Web 应用程序或者网站的配置文件。虽然说 Web.Config 也是个文本文件，但和网页有所不同，用户不能用浏览器浏览 Web.Config 文件。

每张访问数据库的网页都需要与数据库连接，如果把数据库连接信息放到网页上，那么修改数据库连接信息将非常烦琐。因此，通常把数据库连接信息放到 Web.Config 配置文件中。

在 Web.Config 配置文件中设置数据库连接信息，添加语句如下：

```
<connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=.\SQLEXPRESS;
      AttachDbFilename=|DataDirectory|\实例数据库.MDF;Integrated Security=True;User Instance=True"/>
</connectionStrings>
```

需要几点说明：

(1) Data Source 表示 SQL Server 2005 服务器名称，.\SQLEXPRESS 是本地 SQL Server 2005 Express 版默认的服务器名称。

(2) AttachDbFilename 表示数据库的路径和文件名。

(3) |DataDirectory|表示网站默认数据库路径。

12.2.2 访问数据库公共类

本章实例编写了一个 BaseClass.cs 类，负责数据库数据的操作。

1. BaseClass.cs 类的创建

在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【添加新项】命令，在弹出的对话框中选择【类】模板，更改名称为 BaseClass .cs，如图 12-3 所示。



图 12-3 创建 BaseClass.cs 类对话框

2. BaseClass.cs 的主要代码及其解释

(1) BaseClass 类被包含在 GROUP.Manage 命名空间中，以后需要使用 BaseClass 类的页面，必须在页面开头使用 using GROUP.Manage 语句引用命名空间。类结构代码如下：

```
namespace GROUP.Manage
{ //命名空间开始
        public class BaseClass: System.Web.UI.Page
        { //类定义开始
            String strConn; //类变量
            public BaseClass()   //构造函数
            {
                //在构造函数中，取数据库连接串
                strConn = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
            }
            …几个方法的定义
        } //类定义结束
 } //命名空间结束
```

(2) 方法 public DataTable ReadTable(String strSql)用来从数据库读取数据，并返回一个 DataTable。代码如下：

```
public DataTable ReadTable(String strSql)
{
    //创建一个 DataTable
    DataTable dt=new DataTable();
    //定义新的数据连接控件并初始化
    SqlConnection Conn = new SqlConnection(strConn);
    //打开连接
    Conn.Open();
    //定义并初始化数据适配器
    SqlDataAdapter Cmd = new SqlDataAdapter(strSql, Conn);
    //将数据适配器中的数据填充到 DataTable 中
    Cmd.Fill(dt);
    //关闭连接
    Conn.Close();
    //方法返回参数为 DataTable
    return dt;
}
```

(3) 方法 public DataSet ReadDataSet(String strSql)也是用来从数据库读取数据，不同的是返回一个 DataSet。代码如下：

```
public DataSet ReadDataSet(String strSql)
{
    //创建一个数据集 DataSet
    DataSet ds=new DataSet();
    //定义新的数据连接控件并初始化
    SqlConnection Conn = new SqlConnection(strConn);
    //打开连接
    Conn.Open();
    //定义并初始化数据适配器
    SqlDataAdapter Cmd = new SqlDataAdapter(strSql, Conn);
    //将数据填充到数据集 DataSet 中
    Cmd.Fill(ds);
    //关闭连接
    Conn.Close();
    //方法返回参数为 DataSet
    return ds;
}
```

(4) 方法 public DataSet GetDataSet(String strSql, String tableName)和 ReadDataSet 几乎完全相同，只是多了个 tableName 参数。代码如下：

```
public DataSet GetDataSet(String strSql, String tableName)
{
    //创建一个数据集 DataSet
```

```
        DataSet ds = new DataSet();
        //定义新的数据连接控件并初始化
        SqlConnection Conn = new SqlConnection(strConn);
        //打开连接
        Conn.Open();
        //定义并初始化数据适配器
        SqlDataAdapter Cmd = new SqlDataAdapter(strSql, Conn);
        //将数据填充到数据集 DataSet 中
        Cmd.Fill(ds, tableName);
        //关闭连接
        Conn.Close();
        //方法返回参数为 DataSet
        return ds;
    }
```

(5) 方法 public SqlDataReader readrow(String sql)执行 SQL 查询，并返回一个 Reader。代码如下：

```
    public SqlDataReader readrow(String sql)
    {
        //连接数据库
        SqlConnection Conn = new SqlConnection(strConn);
        Conn.Open();
        //定义并初始化 Command 控件
        SqlCommand Comm = new SqlCommand(sql, Conn);
        //创建 Reader 控件，并添加数据记录
        SqlDataReader Reader = Comm.ExecuteReader();
        //如果 Reader 不为空,返回 Reader，否则返回 null
        if (Reader.Read())
        {
            Comm.Dispose();
            return Reader;
        }
        else
        {
            Comm.Dispose();
            return null;
        }
    }
```

(6) 方法 public string Readstr(String strSql, int flag)返回查询结果第 1 行某一字段的值。代码如下：

```
    public string Readstr(String strSql, int flag)
    {
```

```
//创建一个数据集 DataSet
DataSet ds=new DataSet();
String str;
//定义新的数据连接控件并初始化
SqlConnection Conn = new SqlConnection(strConn);
//打开连接
Conn.Open();
//定义并初始化数据适配器
SqlDataAdapter Cmd = new SqlDataAdapter(strSql, Conn);
//将数据填充到数据集 DataSet 中
Cmd.Fill(ds);
// 取出 DataSet 中第一行第“flag”列的数据
str=ds.Tables[0].Rows[0].ItemArray[flag].ToString();
 //关闭连接
 Conn.Close();
//返回数据
return str;
}
```

(7) 方法 public void execsql(String strSql)用来执行 SQL 更新语句。代码如下：

```
public void execsql(String strSql)
{
    //定义新的数据连接控件并初始化
    SqlConnection Conn = new SqlConnection(strConn);
    //定义并初始化 Command 控件
    SqlCommand Comm = new SqlCommand(strSql, Conn);
    //打开连接
    Conn.Open();
    //执行命令
    Comm.ExecuteNonQuery();
    //关闭连接
     Conn.Close();
}
```

12.2.3　母版页

添加母版页，名称为 MasterPage.master。在母版页中添加一个 ScriptManager 控件，这是很重要的。因为很多页面用到 ASP.NET AJAX 无页面刷新技术，直接把该控件放到母版页中，其他用到该母版页的页面就不需要单独添加 ScriptManager 控件了。

母版页上有几个主要的 div，分别设置标题图片、导航、内容和底部信息。新建一个样式文件 StyleSheet.css，定义网站主要样式。母版页设计的最终效果如图 12-4 所示。

部分 HTML 代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
```

```
</asp:ScriptManager>
<div id="maindiv">
<div id="HeadDiv"> <br /> <br /> <br /> <br /> <br />
    您是第<strong style="font-size: 14pt; color: #ffcc66;"><%=Application["counter"]%></strong>位访
问者！  <br />   </div>
<div id="MenuDiv">
    | <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/Default.aspx">首页
      </asp:HyperLink>
    | <asp:HyperLink ID="HyperLink3" runat="server" NavigateUrl="~/about.aspx">关于公司
      </asp:HyperLink>
    | <asp:HyperLink ID="HyperLink4" runat="server" NavigateUrl="~/shownews.aspx?id=%">新闻
        </asp:HyperLink>
    | <asp:HyperLink ID="HyperLink5" runat="server" NavigateUrl="~/showpros.aspx?id=%">产品
        </asp:HyperLink>
    | <asp:HyperLink ID="HyperLink6" runat="server" NavigateUrl="~/address.aspx">联系我们
        </asp:HyperLink> |   </div>
<div id="ContentDiv" style="background-color: #ffffff;">

    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>      </div>
    <div id="EndimageDiv">     </div>
     <div id="EndDiv">
       <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/admin_default.aspx"
Target="_blank">管理入口</asp:HyperLink><br />
              CopyRight &copy; 2008-2009 xingkongsoft All Right Reserved.<br />
              星空软件研究室  版权所有  E-mail:xingkongsoft@163.com
     </div>
 </div>
```



图 12-4　母版页设计页面

该网站只设计一个母版页，在实际工作中，可以根据需要为不同的栏目设计各自的母版页，展现不同的栏目个性。

12.2.4　默认主页

该网站默认主页 Default.aspx，如图 12-5 所示。



图 12-5　默认主页

1. Default.aspx 的 HTML 主要代码

Default.aspx 主要控件包括：用于显示行业和企业新闻的 2 个 GridView 控件，1 个展现企业产品的 DataList 控件，还有用于用户登录和注册的用户控件。主要代码如下：

```
<div    class="divtabletop" style="width: 356px;height: 19px" > ::企业新闻
    <asp:HyperLink ID="HyperLink3" runat="server" NavigateUrl="shownews.aspx?id=企业新闻
        ">More>></asp:HyperLink></div>
<div   class="divtablebody"   style="width: 356px; height: 135px">
   <asp:GridView ID="GridView1" runat="server" Height="131px" PageSize="6" ShowHeader="False"
       Width="336px" GridLines="None" AutoGenerateColumns="False"
       Font-Overline="False" CssClass="font" Font-Italic="False">
     <Columns>
       <asp:HyperLinkField DataNavigateUrlFields="流水号"
              DataNavigateUrlFormatString="shownew.aspx?id={0}"
              DataTextField="新闻标题" DataTextFormatString="&#183;{0}">
              <ItemStyle Font-Overline="False" HorizontalAlign="Left" />
       </asp:HyperLinkField>
       <asp:BoundField DataField="添加时间" DataFormatString="{0:d}" />
     </Columns>
   </asp:GridView>     </div>
...
<div   class="divtabletop" style="width: 357px;height:19px" >::行业新闻
```

```
            <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="shownews.aspx?id=业内新闻
                  ">More>></asp:HyperLink> </div>
<div class="divtablebody"   style="width: 357px;height:135px" >
      <asp:GridView ID="GridView2" runat="server" Height="131px" PageSize="6"
         ShowHeader="False" Width="336px" GridLines="None" AutoGenerateColumns="False"
          CssClass="font">
       <Columns>
          <asp:HyperLinkField DataNavigateUrlFields="流水号"
                DataNavigateUrlFormatString="shownew.aspx?id={0}"
                DataTextField="新闻标题" DataTextFormatString="&#183;{0}">
                <ItemStyle Font-Overline="False" HorizontalAlign="Left" />
          </asp:HyperLinkField>
          <asp:BoundField DataField="添加时间" DataFormatString="{0:d}" />
       </Columns>
      </asp:GridView>   </div>
    ...
<div class="divtabletop" style="width:524px; height: 19px" >::企业产品
    <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="showpros.aspx?id=%">More>>
    </asp:HyperLink>   </div>
<div   class="divtablebody" style="width:524px;height: 265px" >
    <asp:DataList ID="DataList1" runat="server" Height="248px" RepeatColumns="2"
      RepeatDirection="Horizontal"   Width="512px" Font-Names="宋体" Font-Size="12px">
      <ItemTemplate>
      <table border="0" cellpadding="0" cellspacing="0" style="font-size: 12px; font-family: 宋体" >
      <tr> <td align="center" rowspan="2" valign="middle" >
       <a href='showpro.aspx?id=<%# DataBinder.Eval(Container.DataItem, "流水号")%> '>
       <img height="60" src='image/<%# DataBinder.Eval(Container.DataItem, "产品图片")%>'
        width="100" style="border-top-style: none; border-right-style: none; border-left-style: none;
          border-bottom-style: none" alt="a" /></a></td>
     <td valign="middle" style="width: 150px; height: 22px;" align="left">
      <img height="15" src="image/dot_1.gif" style="width: 25px" alt="d" /><a
       href='showpro.aspx?id=<%# DataBinder.Eval(Container.DataItem, "流水号")%> '><strong><%#
       DataBinder.Eval(Container.DataItem, "产品名称")%></strong></a></td> </tr>
     <tr>   <td style="width: 150px; height: 53px" align="left">
     <img height="11" src="image/dot_1.gif" width="24" alt="b" />价格:   ￥<%#
       DataBinder.Eval(Container.DataItem, "产品价格")%>  元<br /> <br />
       <img height="11" src="image/dot_1.gif" width="24" alt="c" />类别:
       <a href='showpros.aspx?id=<%# DataBinder.Eval(Container.DataItem, "产品类别")%>'>
       <%# DataBinder.Eval(Container.DataItem, "产品类别")%> </a> </td> </tr>
     </table>
  </ItemTemplate>
```

```
</asp:DataList> </div>
    ...
<uc1:Userlogin id="Userlogin1_1" runat="server"></uc1:Userlogin>
```

2. Default.aspx.cs 的主要代码及其解释

(1) 创建公共类 BaseClass 的对象，目的是使用操作数据库的方法。

```
BaseClass BaseClass1 = new BaseClass();
```

(2) 每次加载时显示企业新闻、业内新闻和产品信息。

```
protected void Page_Load(object sender, EventArgs e)
{
    string strsql;
    //定义查询企业新闻 SQL 语句，返回前 6 条记录
    strsql = "SELECT top 6  流水号,新闻标题,添加时间   FROM  新闻信息  where  新闻类别='企业新闻' order by  流水号  desc ";
    //把结果返回到 DataTable 中
    DataTable dt = BaseClass1.ReadTable(strsql);
    //指定 GridView 数据源
    GridView1.DataSource = dt;
    // GridView 显示数据
    GridView1.DataBind();
    //定义查询业内新闻 SQL 语句，返回前 6 条记录
    strsql = "SELECT top 6  流水号,新闻标题,添加时间   FROM  新闻信息  where  新闻类别='业内新闻' order by  流水号  desc ";
    //把结果返回到 DataTable 中
    dt = BaseClass1.ReadTable(strsql);
    //指定 GridView 数据源
    GridView2.DataSource = dt;
    // GridView 显示数据
    GridView2.DataBind();
    //定义查询产品信息 SQL 语句，返回前 4 条记录
    strsql = "select top 4 * from  产品  order by  流水号  ";
    //把结果返回到 DataTable 中
    dt = BaseClass1.ReadTable(strsql);
    //指定 GridView 数据源
    DataList1.DataSource = dt;
    // GridView 显示数据
    DataList1.DataBind();
}
```

12.2.5　新闻列表

单击图 12-5 所示窗口中企业新闻或者业内新闻的 More>>链接，进入 shownews.aspx 页面，显示全部的企业新闻或者业内新闻。效果如图 12-6 所示。



图 12-6　新闻列表

1. shownews.aspx 的 HTML 主要代码

页面使用 GridView 控件显示新闻列表。

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" GridLines="None"
              Height="121px" PageSize="6" ShowHeader="False" Width="452px">
<Columns>
  <asp:HyperLinkField DataNavigateUrlFields="流水号"
      DataNavigateUrlFormatString="shownew.aspx?id={0}"
      DataTextField="新闻标题" DataTextFormatString="&#183;{0}" HeaderText="新闻标题">
       <ItemStyle Font-Overline="False" HorizontalAlign="Left" />
  </asp:HyperLinkField>
  <asp:BoundField DataField="添加时间" HeaderText="添加时间" />
  <asp:BoundField DataField="新闻类别" HeaderText="新闻类别" />
  <asp:BoundField DataField="阅读次数" HeaderText="阅读次数" />
</Columns>
</asp:GridView> <br />
当前页码为:[<asp:Label ID="LabelPage" runat="server" Text="1"></asp:Label>] 总页码为:
[<asp:Label ID="LabelTotalPage" runat="server" Text=""></asp:Label>]
<asp:LinkButton   ID="LinkButtonFirst" runat="server" OnClick="LinkButtonFirst_Click">首页
</asp:LinkButton>  
<asp:LinkButton   ID="LinkButtonPrev" runat="server" OnClick="LinkButtonPrev_Click">上一页
</asp:LinkButton>  
```

```
    <asp:LinkButton  ID="LinkButtonNext" runat="server" OnClick="LinkButtonNext_Click">下一页
</asp:LinkButton>  
    <asp:LinkButton  ID="LinkButtonLast" runat="server" OnClick="LinkButtonLast_Click">末页
</asp:LinkButton>
```

2. shownews.aspx.cs 的主要代码及其解释

```
    //创建公共类 BaseClass 的对象，目的是使用操作数据库的方法。
    BaseClass BaseClass1 = new BaseClass();
    //每次加载时显示新闻。
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack) getGoods();
    }
    private void getGoods()
    {
        //获取数据，入口参数 Request.Params["id"].ToString()为“%”表示全部新闻，为“业内新闻”
表示行业新闻，为“企业新闻”表示企业新闻
        string strsql = "select  * from  新闻信息  where  新闻类别  like '" + Request.Params["id"].ToString()
+ "' order by  流水号  desc";
        DataTable dt = BaseClass1.ReadTable(strsql);
        //实现分页
        PagedDataSource objPds = new PagedDataSource();
        objPds.DataSource = dt.DefaultView;
        objPds.AllowPaging = true;
        objPds.PageSize = 12;
        int CurPage = Convert.ToInt32(this.LabelPage.Text);
        objPds.CurrentPageIndex = CurPage - 1;
        if (objPds.CurrentPageIndex < 0)
        {
            objPds.CurrentPageIndex = 0;
        }
        //只有一页时禁用上页、下页按钮
        if (objPds.PageCount == 1)
        {
            LinkButtonPrev.Enabled = false;
            LinkButtonNext.Enabled = false;
        }
        else//多页时
        {
            //为第一页时
            if (CurPage == 1)
```

```
            {
                LinkButtonPrev.Enabled = false;
                LinkButtonNext.Enabled = true;
            }
            //是最后一页时
            if (CurPage == objPds.PageCount)
            {
                LinkButtonPrev.Enabled = true;
                LinkButtonNext.Enabled = false;
            }
        }
        this.LabelTotalPage.Text = Convert.ToString(objPds.PageCount);
        GridView1.DataSource = objPds;
        GridView1.DataBind();
    }
    //首页
    protected void LinkButtonFirst_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = "1";
        getGoods();
    }
    //上一页
    protected void LinkButtonPrev_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = Convert.ToString(int.Parse(this.LabelPage.Text) - 1) ;
        getGoods();
    }
    //下一页
    protected void LinkButtonNext_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = Convert.ToString(int.Parse(this.LabelPage.Text) + 1) ; ;
        getGoods();
    }
    //末页
    protected void LinkButtonLast_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = this.LabelTotalPage.Text;
        getGoods();
    }
```

12.2.6　产品列表

单击图 12-5 所示窗口中企业产品栏目的 More>>链接，进入 showpros.aspx 页面。产品列表效果如图 12-7 所示。



图 12-7　产品列表

1. showpros.aspx 的 HTML 主要代码

页面使用 DataList 控件显示产品列表。

```
<asp:DataList ID="DataList1" runat="server" Height="200px"
              OnSelectedIndexChanged="DataList1_SelectedIndexChanged1"
              RepeatColumns="2" RepeatDirection="Horizontal" Width="532px">
<ItemTemplate>
<table border="0" cellpadding="0" cellspacing="0">
 <tr> <td align="center" rowspan="2" valign="middle">
 <a href='showpro.aspx?id=<%# DataBinder.Eval(Container.DataItem, "流水号")%> '>
 <img alt="a" height="60" src='image/<%# DataBinder.Eval(Container.DataItem, "产品图片")%>'
style="border-top-style: none; border-right-style: none; border-left-style: none;   border-bottom-style: none"
width="100" /></a></td>
 <td align="left" style="width: 150px; height: 22px;" valign="middle">
 <img alt="d" height="15" src="image/dot_1.gif" style="width: 25px" /><a href='showpro.aspx?id=<%#
DataBinder.Eval(Container.DataItem, "流水号")%> '><strong><%# DataBinder.Eval(Container.DataItem, "产
品名称")%></strong></a></td> </tr>
<tr> <td align="left" style="width: 150px; height: 53px">
 <img alt="b" height="11" src="image/dot_1.gif" width="24" />价格：  ￥<%#
DataBinder.Eval(Container.DataItem, "产品价格")%>元<br />
<img alt="c" height="11" src="image/dot_1.gif" width="24" />类别：
<a href='showpros.aspx?id=<%# DataBinder.Eval(Container.DataItem, "产品类别")%>'>
      <%# DataBinder.Eval(Container.DataItem, "产品类别")%> </a> </td> </tr>
```

```
    </table>
    </ItemTemplate>
    </asp:DataList><br />
    当前页码为:[<asp:Label ID="LabelPage" runat="server" Text="1"></asp:Label>] 总页码为:
[<asp:Label ID="LabelTotalPage" runat="server" Text=""></asp:Label>]
    <asp:LinkButton   ID="LinkButtonFirst" runat="server" OnClick="LinkButtonFirst_Click">首页
</asp:LinkButton>  
    <asp:LinkButton   ID="LinkButtonPrev" runat="server" OnClick="LinkButtonPrev_Click">上一页
</asp:LinkButton>  
    <asp:LinkButton   ID="LinkButtonNext" runat="server" OnClick="LinkButtonNext_Click">下一页
</asp:LinkButton>  
    <asp:LinkButton   ID="LinkButtonLast" runat="server" OnClick="LinkButtonLast_Click">末页
</asp:LinkButton>
```

2. showpros.aspx.cs 的主要代码及其解释

```
    //创建公共类 BaseClass 的对象，目的是使用操作数据库的方法。
    BaseClass BaseClass1 = new BaseClass();
    //每次加载时显示新闻。
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack) getGoods();
    }
    private void getGoods()
    {
        //获取数据 入口参数 Request.Params["id"].ToString()为“%”表示全部产品，否则为具体类型
        string strsql = "select   * from 产品 where 产品类别 like '" + Request.Params["id"].ToString() + "'
order by   流水号";
        DataTable dt = BaseClass1.ReadTable(strsql);
        //实现分页
        PagedDataSource objPds = new PagedDataSource();
        objPds.DataSource = dt.DefaultView;
        objPds.AllowPaging = true;
        objPds.PageSize =8;
        int CurPage = Convert.ToInt32(this.LabelPage.Text);
        objPds.CurrentPageIndex = CurPage - 1;
        if (objPds.CurrentPageIndex < 0)
        {
            objPds.CurrentPageIndex = 0;
        }
        //只有一页时禁用上页、下页按钮
        if (objPds.PageCount == 1)
        {
            LinkButtonPrev.Enabled = false;
```

```
            LinkButtonNext.Enabled = false;
        }
        else//多页时
        {
            //为第一页时
            if (CurPage == 1)
            {
                LinkButtonPrev.Enabled = false;
                LinkButtonNext.Enabled = true;
            }
            //是最后一页时
            if (CurPage == objPds.PageCount)
            {
                LinkButtonPrev.Enabled = true;
                LinkButtonNext.Enabled = false;
            }
        }
        this.LabelTotalPage.Text = Convert.ToString(objPds.PageCount);
        DataList1.DataSource = objPds;
        DataList1.DataBind();
    }
    //首页
    protected void LinkButtonFirst_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = "1";
        getGoods();
    }
    //上一页
    protected void LinkButtonPrev_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = Convert.ToString(int.Parse(this.LabelPage.Text) - 1) ;
        getGoods();
    }
    //下一页
    protected void LinkButtonNext_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = Convert.ToString(int.Parse(this.LabelPage.Text) + 1) ; ;
        getGoods();
    }
    //末页
    protected void LinkButtonLast_Click(object sender, EventArgs e)
    {
        this.LabelPage.Text = this.LabelTotalPage.Text;
        getGoods();
    }
```

12.2.7　用户登录用户控件

方便起见，将用户登录对话框做成了用户控件，如图 12-8 所示。用户登录后出现右边信息，系统为注册用户提供了订单管理等功能。

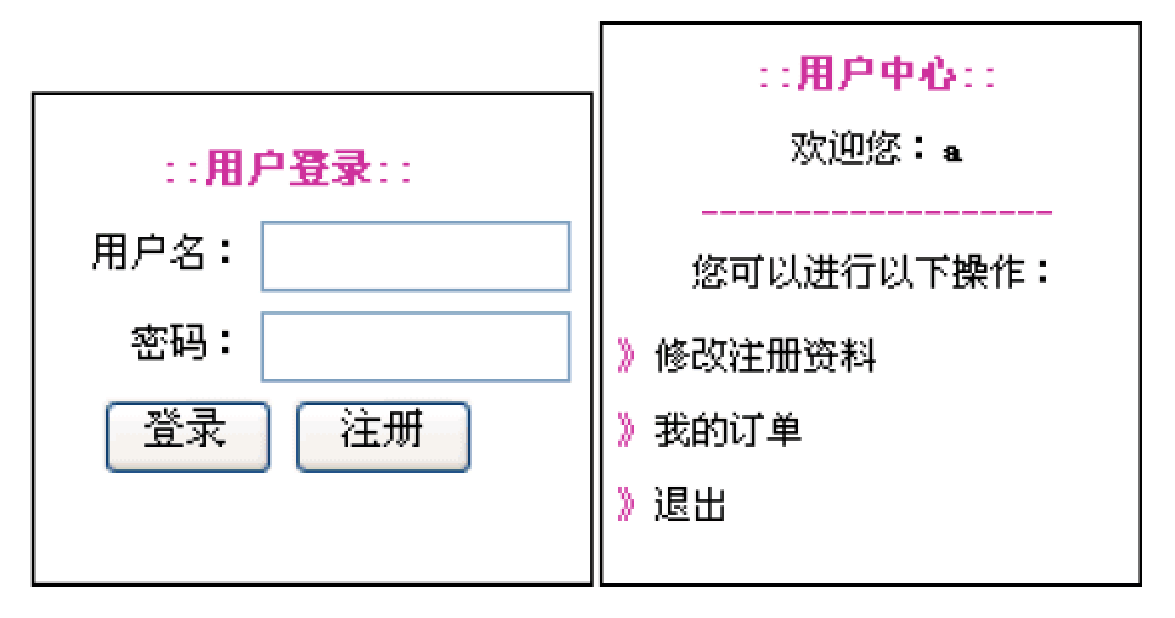


图 12-8　用户登录对话框

1. Userlogin.ascx 的 HTML 主要代码

该用户控件采用了上下 2 个 div 层，分别存放如图 12-8 所示左边和右边的信息。通过 div 的 Visible 属性控制显示内容。

```
<div id="div1" runat="server"   style="width: 100%; height: 100px;">
<table style="font-size: 12px; font-family: 宋体;">
  <tr> <td colspan="2" style="width: 180px; height: 21px;" align="center"> ::用户登录::</td> </tr>
  <tr> <td style="width: 80px" align="right"> 用户名：</td>
        <td style="width: 83px"> <asp:TextBox ID="TextBox1" runat="server"
Width="90"></asp:TextBox></td> </tr>
  <tr> <td style="width: 80px" align="right">  密码：</td>
        <td style="width: 83px"> <asp:TextBox ID="TextBox2" runat="server" Width="90"
TextMode="Password"></asp:TextBox></td> </tr>
  <tr> <td style="width: 180px" colspan="2" align="center">
         <asp:Button ID="Button1" runat="server" Text="登录" Width="53px"
OnClick="Button1_Click" />
         <asp:Button ID="Button2" runat="server" Text="注册" Width="56px"
OnClick="Button2_Click" /></td> </tr>
</table>
</div>
<div id="div2" runat="server"   style="width: 100%; height: 130px; ">
  <table style="width: 100% ;font-size: 12px; font-family: 宋体;">
   <tr> <td style="width: 180px" align="center"> ::用户中心::</td>   </tr>
   <tr> <td style="width: 180px; height: 55px;" align="center"> 欢迎您：<asp:Label ID="Label1"
runat="server">Label</asp:Label><br /> <br /> 您可以进行以下操作：</td>   </tr>
   <tr> <td style="width: 120px; height: 89px; text-align: center; " align="center">
       <table style="font-size: 12px; font-family: 宋体;">
       <tr> <td style="width: 120px" align="left"> 》 <a href="useredit.aspx">修改注册资料</a></td>
       </tr>
       <tr> <td style="width: 120px; height: 20px;" align="left"> 》<a href="userorder.aspx">我的订单
```

```
</a></td> </tr>
              <tr> <td style="width: 120px; height: 20px;" align="left"> 》 <a href="exit.aspx">退出</a></td>
            </tr> </table>
            </td> </tr>
        </table>
        </div>
```

2. Userlogin.ascx.cs 的主要代码及其解释

(1) 创建公共类 BaseClass 的对象，目的是使用其中数据库操作的方法。

```
BaseClass BaseClass1 = new BaseClass();
```

(2) 加载是判断用户是否登录，决定显示如图 12-8 所示左边或者右边的信息。

```
protected void Page_Load(object sender, EventArgs e)
{
   div1.Visible = false;
   div2.Visible = false;
   if (Session["name"] != null)
   {
         Label1.Text = Session["name"].ToString();
         div2.Visible = true;
   }
   else
   {
         div1.Visible = true;
   }
}
```

(3) 单击【登录】按钮触发 Button1_Click 事件。

```
protected void Button1_Click(object sender, EventArgs e)
{
      //管理员标志=0，表示普通用户。管理员标志=1，表示管理员。
      string strsql = "select * from 用户 where 管理员标志=0 and 用户名 ='" + TextBox1.Text + "' and 
密码 ='" + TextBox2.Text + "'";
      DataSet ds = new DataSet();
      ds = BaseClass1.GetDataSet(strsql, "username");
      if (ds.Tables["username"].Rows.Count == 0)
      {
            string scriptString = "alert('" + "用户名不存在或密码错误，请确认后再登录！" + "');";
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", scriptString, true);
      }
      else
      {
```

```
                Session["name"] = TextBox1.Text;
                Label1.Text = "<b>" + Session["name"].ToString() + "</b>";
                div1.Visible = false;
                div2.Visible = true;
            }
        }
```

(4) 单击【注册】按钮触发 Button2_Click 事件。

```
protected void Button2_Click(object sender, EventArgs e)
{
    Response.Write("<script>window.location='userreg.aspx';</script>");
}
```

12.2.8　用户注册页面

单击图 12-8 所示的用户登录对话框的【注册】按钮，进入注册用户信息页面，如图 12-9 所示。

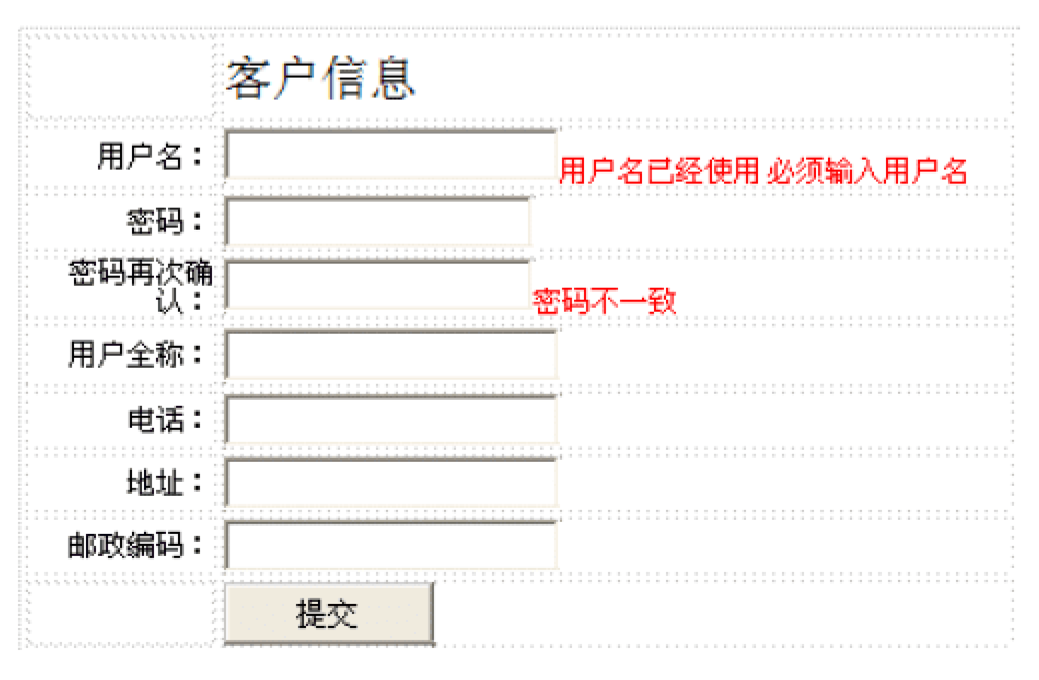


图 12-9　用户注册对话框

1. Userlogin.ascx 的 HTML 主要代码

页面使用了 3 个验证控件 RequiredFieldValidator、CustomValidator 和 CompareValidator。RequiredFieldValidator 和 CustomValidator 控制用户名不能为空，并且不能已经使用。CompareValidator 验证控件用来比较第一次输入的密码和再次确认密码是否一致。

```
<table style="width: 413px">
  <tr> <td style="width: 100px; height: 36px;"> </td>
      <td style="width: 369px; font-size: 20px; height: 36px;" align="left">客户信息</td> </tr>
  <tr> <td style="width: 100px" align="right">    用户名：</td>
      <td style="width: 369px" align="left">
         <asp:TextBox ID="TextBox1" runat="server" Width="139px"></asp:TextBox>
         <asp:CustomValidator ID="CustomValidator1" runat="server" ControlToValidate="TextBox1"
              ErrorMessage="用户名已经使用" OnServerValidate="CustomValidator1_ServerValidate"
ValidateEmptyText="True" Display="Dynamic" Width="86px"></asp:CustomValidator>
         <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="必
```

```
须输入用户名" ControlToValidate="TextBox1"></asp:RequiredFieldValidator></td> </tr>
     <tr> <td style="width: 100px" align="right"> 密码：</td>
          <td style="width: 369px" align="left">
      <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox></td> </tr>
     <tr> <td style="width: 100px" align="right">  密码再次确认：</td>
          <td style="width: 369px" align="left">
            <asp:TextBox ID="TextBox3" runat="server" TextMode="Password"></asp:TextBox>
            <asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="TextBox2"
       ControlToValidate="TextBox3" ErrorMessage="密码不一致"></asp:CompareValidator></td></tr>
      <tr> <td style="width: 100px; height: 26px;" align="right"> 用户全称：</td>
            <td style="width: 369px; height: 26px;" align="left">
              <asp:TextBox ID="TextBox4" runat="server" Width="139px"></asp:TextBox></td> </tr>
      <tr> <td style="width: 100px" align="right">电话：</td>
            <td style="width: 369px" align="left">
             <asp:TextBox ID="TextBox5" runat="server" Width="139px"></asp:TextBox></td> </tr>
      <tr> <td style="width: 100px; height: 21px" align="right">  地址：</td>
            <td style="width: 369px; height: 21px" align="left">
             <asp:TextBox ID="TextBox6" runat="server" Width="139px"></asp:TextBox></td> </tr>
      <tr> <td style="width: 100px" align="right"> 邮政编码：</td>
            <td style="width: 369px" align="left">
            <asp:TextBox ID="TextBox7" runat="server" Width="139px"></asp:TextBox></td> </tr>
      <tr> <td style="width: 100px"> </td>
            <td style="width: 369px" align="left">
     <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="提交" Width="87px"
/></td> </tr> </table>
```

2. Userlogin.ascx.cs 的主要代码及其解释

(1) 创建公共类 BaseClass 的对象，目的是使用数据库操作的方法。

```
BaseClass BaseClass1 = new BaseClass();
```

(2) 验证用户名是否已经使用，触发 CustomValidator1_ServerValidate 事件。

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    //args.Value 为需要验证的用户名
    string strsql = "select * from  用户  where  用户名  ='" + args.Value.ToString() + "'";
    DataSet ds = new DataSet();
    ds = BaseClass1.GetDataSet(strsql, "username");
    // args.IsValid 是否通过验证的返回值
    if (ds.Tables["username"].Rows.Count > 0)
    {
        args.IsValid = false;
    }
    else
```

```
        {
            args.IsValid = true;
        }
    }
```

(3) 单击【提交】按钮触发 Button1_Click 事件。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
      if (CustomValidator1.IsValid == true)
        {
            string strsql;
            strsql = "insert into 用户 (用户名,密码,真实姓名,电话,地址,邮编) values ('" + TextBox1.Text
+ "','" + TextBox2.Text + "','" + TextBox4.Text + "','" + TextBox5.Text + "','" + TextBox6.Text + "','" +
TextBox7.Text + "')";
            BaseClass1.execsql(strsql);
            Response.Write("<script>alert(\"注册成功！\");</script>");
            Session["name"] = TextBox1.Text;
            Response.Redirect("Default.aspx");
        }
    }
```

12.2.9　产品订单

当单击图 12-7 所示的页面中的产品标题或产品图片时，显示如图 12-10 所示的产品详细信息。如果用户已经登录，单击图 12-10 中的【订购>>】链接时，出现产品订单页面，如图 12-11 所示。

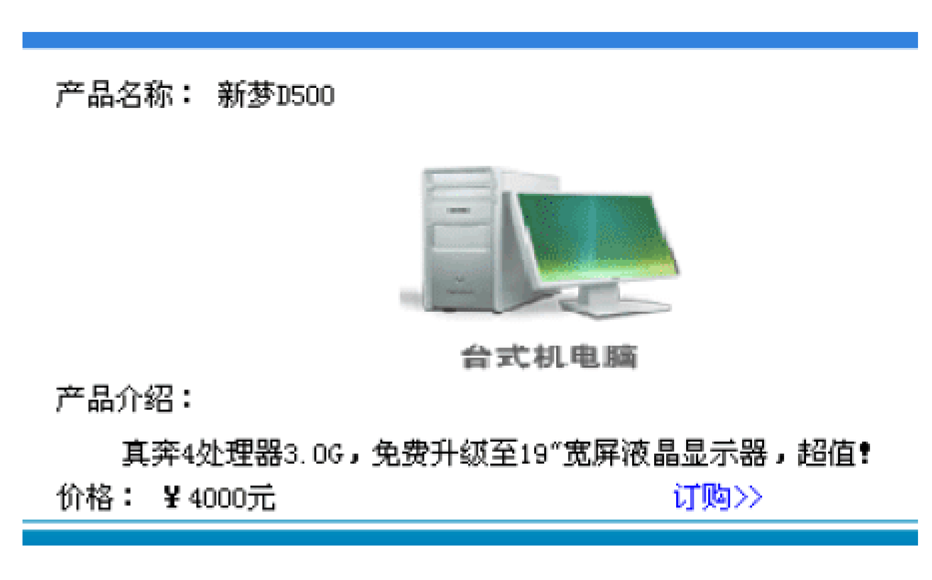


图 12-10　产品详细信息

订购信息

产品名称： 新梦D500

单价： 4000

订购数量： 1

提交订单

图 12-11　产品订单

1. order.aspx 的 HTML 主要代码

```
<table>
<tr> <td style="width: 134px; height: 36px"> </td>
     <td align="left" style="width: 220px; height: 36px">  订购信息</td> </tr>
<tr> <td align="right" style="width: 134px; height: 33px"> 产品名称：</td>
     <td align="left" style="width: 220px; height: 33px">
  <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label></td> </tr>
<tr> <td align="right" style="width: 134px; height: 30px"> 单价：</td>
     <td style="width: 220px; height: 30px" align="left">
  <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label></td> </tr>
<tr> <td align="right" style="width: 134px; height: 36px">  订购数量：</td>
     <td style="width: 220px; height: 36px" align="left">
   <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td></tr>
<tr> <td style="width: 134px; height: 38px"></td>
     <td align="left" style="width: 220px; height: 38px">
<asp:Button ID="Button1" runat="server" Text="提交订单" OnClick="Button1_Click" /></td></tr>
</table>
```

2. order.aspx.cs 的主要代码及其解释

(1) 创建公共类 BaseClass 的对象，目的是使用操作数据库的方法。

```
BaseClass BaseClass1 = new BaseClass();
```

(2) 如果用户已登录，输入订货数量，否则提示用户登录。

```
protected void Page_Load(object sender, EventArgs e)
{
    // 判断用户是否登录
    if (Session["name"] == null)
    {
        Response.Write("<script>alert(\"请登录！\");</script>");
        Response.Redirect("default.aspx");
    }
    // 首次加载初始化
    if (!Page.IsPostBack)
    {
       // Request.QueryString["id"]为页面入口参数，表示所订产品
       string strsql = "select 产品名称,产品价格 from 产品 where 流水号 =" +
Request.QueryString["id"];
       DataTable dt = new DataTable();
       dt = BaseClass1.ReadTable(strsql);
       Label1.Text = dt.Rows[0].ItemArray[0].ToString();
       Label2.Text = dt.Rows[0].ItemArray[1].ToString();
       TextBox1.Text = "1";
```

```
        }
    }
```

(3) 单击【提交订单】按钮时触发 Button1_Click 事件。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        string strsql;
        strsql = "insert into  订单 (产品流水号,订购数量,用户名,订购日期) values (" +
Request.QueryString["id"] + "," + TextBox1.Text + ",'" + Session["name"].ToString() + "',convert(datetime,'" +
DateTime.Today.ToShortDateString() + "',120))";
        BaseClass1.execsql(strsql);
        Response.Write("<script>alert(\"提交成功，您还可以选购其他商品！\");</script>");
        Response.Redirect("showpros.aspx?id=%");
    }
```

12.2.10　管理员登录页面

各个页面的底部几乎都有【管理入口】链接，单击该链接进入管理员的登录页面，如图 12-12 所示。



图 12-12　管理员登录页面

1. login.aspx 的 HTML 主要代码

如图 12-12 所示的登录对话框实际上就是一个 Login 控件，通过调整控件属性达到满意的效果。HTML 代码如下：

```
    <asp:Login ID="Login1" runat="server" BackColor="#EFF3FB" BorderColor="#B5C7DE"
        BorderPadding="4" BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
        Font-Size="0.8em" ForeColor="#333333" Height="180px" Width="275px"
OnAuthenticate="Login1_Authenticate1">
        <TitleTextStyle BackColor="#507CD1" Font-Bold="True" Font-Size="0.9em" ForeColor="White" />
        <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
        <TextBoxStyle Font-Size="0.8em" />
        <LoginButtonStyle BackColor="White" BorderColor="#507CD1" BorderStyle="Solid"
BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em" ForeColor="#284E98" />
```

```
</asp:Login>
```

2. login.aspx.cs 的主要代码及其解释

登录页面的代码比较简单，只在用户登录的时候触发 Login1_Authenticate1 事件，此事件用来判断该用户是否合法。默认管理员名称及密码为 admin。其代码如下：

```
protected void Login1_Authenticate1(object sender, AuthenticateEventArgs e)
{
    //定义 SQL 查询语句
    string strsql = "select * from 用户 where 用户名 ='" + Login1.UserName.ToString() + "' and 密码
='" + Login1.Password.ToString() + "' ";
    //创建 DataTable
    DataTable dt = new DataTable();
    //调用 ReadTable 方法获取查询结果
    dt = BaseClass1.ReadTable(strsql);
    //判断是否有符合条件的记录
    if (dt.Rows.Count > 0)
    {
        //将合法的用户名放到 Session 对象中，表示用户已经登录
        Session["admin"] = Login1.UserName.ToString();
        //跳转到后台管理页面 admin_default.aspx
        Response.Redirect("admin_default.aspx");
    }
}
```

12.2.11　后台管理页面

管理员登录成功后，进入如图 12-13 所示的后台管理页面。该页面提供了新闻管理、新闻添加、产品管理、产品添加和订单管理等功能。

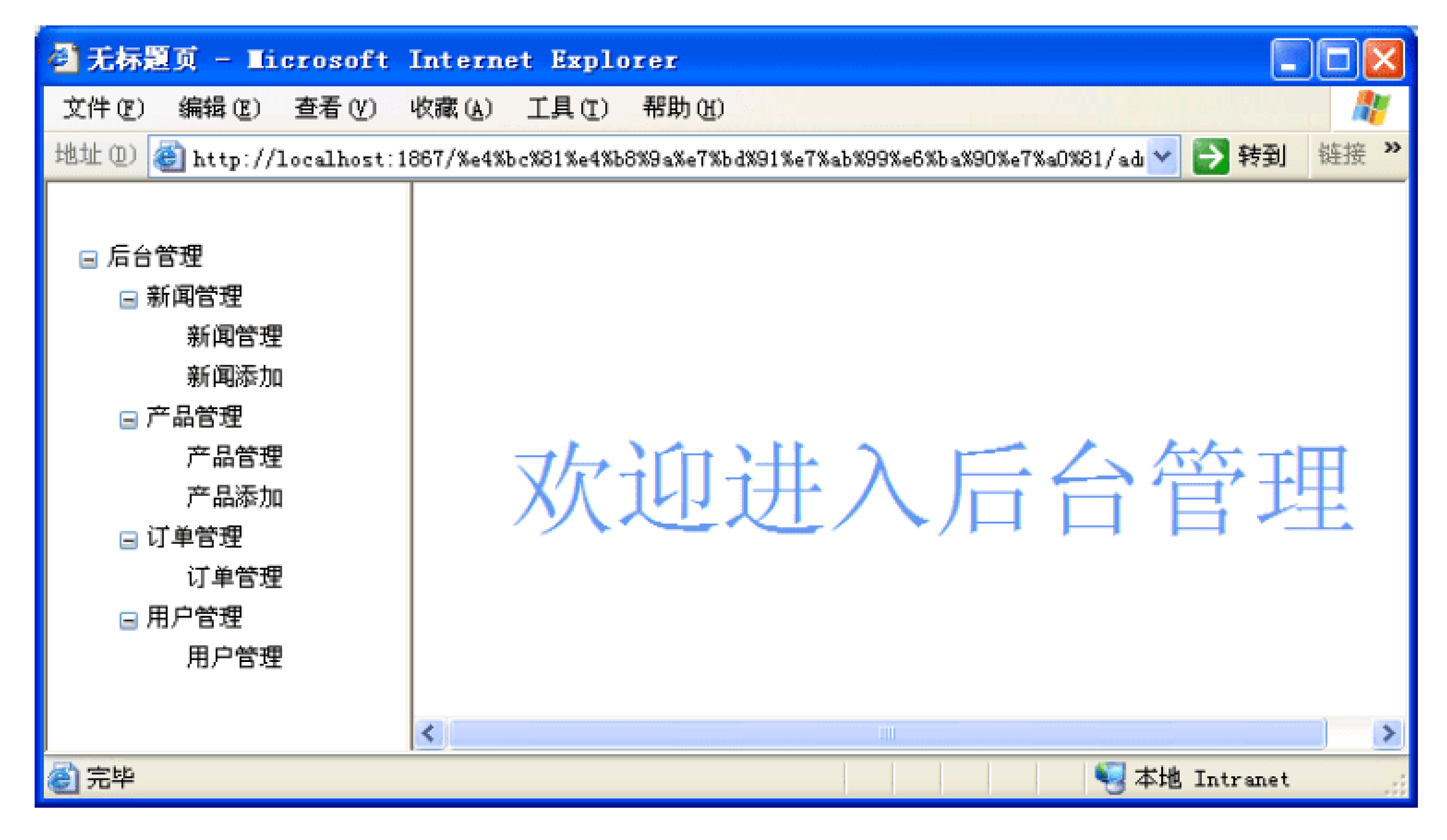


图 12-13　后台管理页面

1. admin_default.aspx 的 HTML 主要代码

页面中添加了 1 个 TreeView 控件和 1 个框架集，TreeView 控件显示管理功能，框架集用于相应管理页面的显示。

```
<asp:TreeView ID="TreeView1" runat="server" Height="264px"
OnSelectedNodeChanged="TreeView1_SelectedNodeChanged"   Width="60px">
  <Nodes>
    <asp:TreeNode Text="后台管理" Value="后台管理">
    <asp:TreeNode Text="新闻管理" Value="新闻管理">
    <asp:TreeNode Text="新闻管理" Value="新闻管理"></asp:TreeNode>
    <asp:TreeNode Text="新闻添加" Value="新闻添加"></asp:TreeNode> </asp:TreeNode>
    <asp:TreeNode Text="产品管理" Value="产品管理">
    <asp:TreeNode Text="产品管理" Value="产品管理"></asp:TreeNode>
    <asp:TreeNode Text="产品添加" Value="产品添加"></asp:TreeNode> </asp:TreeNode>
    <asp:TreeNode Text="订单管理" Value="订单管理">
    <asp:TreeNode Text="订单管理" Value="订单管理"></asp:TreeNode></asp:TreeNode>
    <asp:TreeNode Text="用户管理" Value="用户管理">
    <asp:TreeNode Text="用户管理" Value="用户管理"></asp:TreeNode>
          </asp:TreeNode> </asp:TreeNode>
  </Nodes>
</asp:TreeView>
<iframe style="width: 100%; height: 100%;" id="iframe1" runat="server" frameborder="0">
</iframe>
```

2. admin_default.aspx.cs 的主要代码及其解释

每次后台管理页面加载时检查管理员是否登录，如果没有登录，就跳转到管理员登录页面。

```
protected void Page_Load(object sender, EventArgs e)
{
//判断是否登录?
  if (Session["admin"] == null)
  {
  //跳转到管理员登录页面
        Response.Redirect("login.aspx");
  }
}
```

12.2.12 新闻管理页面

单击如图 12-13 所示的页面中的【新闻管理】链接，进入新闻管理页面，如图 12-14 所示。

新闻标题	新闻类别	阅读次数	添加时间	
・预装Vista系统 华硕A8H52笔记本降2千	[业内新闻]	[3]	2007-1-20 0:00:00	删除
・只谈性价比 神舟近期降价促销本本一览	[业内新闻]	[1]	2007-1-5 0:00:00	删除
・送40G硬盘还降价 惠普nx6330仅9999元	[业内新闻]	[4]	2007-1-20 0:00:00	删除
・联想新品旭日上市 双核仅售6200元	[业内新闻]	[0]	2007-2-7 0:00:00	删除
・将遭抢购 五款人气最旺的低价本推荐	[业内新闻]	[1]	2007-5-2 0:00:00	删除
・不仅仅为了游戏 双核独显笔记本推荐	[业内新闻]	[2]	2007-5-5 0:00:00	删除
下一页				

图 12-14　新闻管理页面

1. delnews.aspx 的 HTML 主要代码

在页面中使用了 GridView 控件，该控件增加了【删除】列，用于删除过期的新闻。代码如下：

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" BackColor="White"
…
<Columns>
<asp:HyperLinkField DataNavigateUrlFields="流水号"
    DataNavigateUrlFormatString="showpro.aspx?id={0}"
    DataTextField="产品名称" DataTextFormatString="&#183;{0}" HeaderText="产品名称"
Target="main">    <ItemStyle HorizontalAlign="Left" />
</asp:HyperLinkField>
<asp:BoundField DataField="产品类别" DataFormatString="[{0}]" HeaderText="产品类别" />
<asp:BoundField DataField="产品价格" DataFormatString="{0}元" HeaderText="产品价格" />
<asp:BoundField DataField="产品图片" HeaderText="产品图片" />
<asp:CommandField ShowCancelButton="False" ShowDeleteButton="True" />
</Columns>
…
</asp:GridView>
```

2. delnews.aspx.cs 的主要代码及其解释

(1) 加载时判断是否已经登录。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)
    {
        // 跳转到登录页面
        Response.Redirect("login.aspx");
    }
    //显示所有新闻
```

```
        bindgrig();
    }
```

(2) 单击【删除】按钮时触发 GridView1_RowDeleting 事件。

```
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        //定义删除语句
    String strsql = "delete from 新闻信息 where 流水号=" +
GridView1.DataKeys[e.RowIndex].Value.ToString() + "";
        //执行 SQL 命令
        BaseClass1.execsql(strsql);
        //重新显示新闻
        bindgrig();
    }
```

(3) bindgrig()是自定义函数，用于检索新闻，显示到 GridView 控件上。

```
    void bindgrig()
    {
        //定义 SQL 检索语句
        string strsql = "select  * from 新闻信息  order by  流水号 ";
      //创建 DataTable，并返回数据
        DataTable dt = BaseClass1.ReadTable(strsql);
      //设置 GridView 数据源
        GridView1.DataSource = dt;
      //显示数据
        GridView1.DataBind();
    }
```

(4) 单击【上一页】、【下一页】按钮时，触发 GridView1_PageIndexChanging 事件。

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex = e.NewPageIndex;
    bindgrig();
}
```

12.2.13　产品添加页面

单击如图 12-13 所示的页面中的【产品添加】链接，进入产品添加页面，如图 12-15 所示。

图 12-15　产品添加页面

1. addpro.aspx 的 HTML 主要代码

主要控件包括：TextBox、FileUpload 和 DropDownList 等控件。代码如下：

```
<strong>添加产品</strong>
…
产品名称 <asp:TextBox ID="TextBox1" runat="server" Width="209px"></asp:TextBox>
…
价格 <asp:TextBox ID="TextBox3" runat="server" Width="209px"></asp:TextBox></td></tr>
图片 <asp:FileUpload ID="FileUpload1" runat="server" />
产品类别 <asp:DropDownList ID="DropDownList1" runat="server" Width="120px">
          </asp:DropDownList>
简介 <asp:TextBox ID="TextBox2" runat="server" Height="150px" TextMode="MultiLine"
Width="300px"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click" /></td> </tr>
```

2. addpro.aspx.cs 的主要代码及其解释

(1) 每次加载时判断是否已经登录，第一次加载初始化产品类别下拉列表框。

```
protected void Page_Load(object sender, System.EventArgs e)
{
    if (Session["admin"] == null)
    {
        Response.Redirect("login.aspx");
    }
    // 判断是否第一次加载
    if (!Page.IsPostBack)
    {
      // 第一次加载初始化下拉列表框
      DataTable dt = new DataTable();
      string strsql = "select * from 产品类别";
```

```
            dt = BaseClass1.ReadTable(strsql);
            DropDownList1.DataSource = dt;
            DropDownList1.DataTextField = "产品类别";
            DropDownList1.DataValueField = "产品类别";
            DropDownList1.DataBind();
        }
    }
```

(2) 单击【提交】按钮时触发 Button1_Click 事件。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
            string strsql;
            //定义 SQL 插入语句
        strsql = "insert into  产品 (产品名称,产品价格,产品图片,产品类别,产品介绍) values ('" +
TextBox1.Text + "','" + TextBox3.Text + "','" + FileUpload1.FileName + "','" + DropDownList1.SelectedValue
+ "','" + TextBox2.Text + "')";
            //执行 SQL 插入语句
            BaseClass1.execsql(strsql);
            //上传产品图片
            if (FileUpload1.HasFile == true)
            {
                FileUpload1.SaveAs(Server.MapPath(("~/image/") + FileUpload1.FileName));
            }
            //提示提交成功
            Response.Write("<script>alert(\"产品添加成功！ \");</script>");
            //清空产品名称、价格、图片和简介文本编辑器
            TextBox1.Text = "";
            TextBox2.Text = "";
            TextBox3.Text = "";
    }
```

12.2.14　订单管理页面

单击如图 12-13 所示的页面中的【订单管理】链接，进入订单管理页面，如图 12-16 所示。订单管理提供两个功能，一个是删除过期订单，另一个是编辑订单的处理标志。

产品号	用户名	订购数量	订购日期			
3	chen	2	2008-8-2 0:00:00	☐是否处理	编辑	删除
8	chen	1	2008-8-2 0:00:00	☐是否处理	编辑	删除
7	chen	1	2008-8-2 0:00:00	☐是否处理	编辑	删除
4	chen	1	2008-8-2 0:00:00	☐是否处理	编辑	删除
1	chen	1	2008-8-2 0:00:00	☐是否处理	编辑	删除
5	aa	1	2008-4-26 0:00:00	☑是否处理	编辑	删除

图 12-16　订单管理页面

1. delorder.aspx 的 HTML 主要代码

该页面采用了 GridView 控件。代码如下：

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AutoGenerateColumns="False"
…
  <Columns>
    <asp:BoundField DataField="产品流水号" HeaderText="产品号" ReadOnly="True" />
    <asp:BoundField DataField="用户名" HeaderText="用户名" ReadOnly="True" />
    <asp:BoundField DataField="订购数量" HeaderText="订购数量" ReadOnly="True" />
    <asp:BoundField DataField="订购日期" HeaderText="订购日期" ReadOnly="True" />
    <asp:CheckBoxField DataField="处理标志" Text="是否处理" />
    <asp:CommandField ShowEditButton="True" />
    <asp:CommandField ShowCancelButton="False" ShowDeleteButton="True" />
  </Columns>
…
</asp:GridView>
```

2. delorder.aspx.cs 的主要代码及其解释

(1) 加载时判断管理员是否已经登录。

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Session["admin"] == null)
  {
       Response.Redirect("login.aspx");
  }
  if (!Page.IsPostBack)
  {
       bindgrig();
   }
}
```

(2) 单击【删除】按钮时触发 GridView1_RowDeleting 事件。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
   String strsql = "delete from  订单  where  流水号=" +
                   GridView1.DataKeys[e.RowIndex].Value.ToString() + "";
   BaseClass1.execsql(strsql);
   bindgrig();
}
```

(3) 在编辑状态下，单击【更新】按钮时触发 GridView1_ RowUpdating 事件。

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
```

```
{
    //提交行修改 (CheckBox)GridView1.Rows[e.RowIndex].FindControl("CheckBox1")
    string str;
    CheckBox ck = (CheckBox)GridView1.Rows[e.RowIndex].Cells[4].Controls[0];
    if (ck.Checked == true)
    {
        str = "1";
    }
    else
    {
        str = "0";
    }
    String strsql = "update  订单 set 处理标志=" + str + " where 流水号=" +
                    GridView1.DataKeys[e.RowIndex].Value.ToString() + "";
    BaseClass1.execsql(strsql);
    GridView1.EditIndex = -1;
    bindgrig();
}
```

(4) 在编辑状态下，单击【取消】按钮时触发 GridView1_ RowCancelingEdit 事件。

```
protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
{
    GridView1.EditIndex = -1;
    bindgrig();
}
```

(5) bindgrig()是自定义函数，将订单显示到 GridView 控件上。

```
void bindgrig()
{
    string strsql = "select  * from  订单 order by  流水号 desc";
    DataTable dt = BaseClass1.ReadTable(strsql);
    GridView1.DataSource = dt;
    GridView1.DataBind();
}
```

(6) 单击【上一页】、【下一页】按钮时，触发 GridView1_PageIndexChanging 事件。

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex = e.NewPageIndex;
    bindgrig();
}
```

(7) 单击【编辑】按钮时触发 GridView1_ RowEditing 事件。

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex = e.NewEditIndex;
    bindgrig();
}
```

12.2.15　用户管理页面

单击如图 12-13 所示的页面中的【用户管理】链接，进入用户管理页面，如图 12-17 所示。

用户名	真实姓名	电话	地址	邮编	
a	试验				删除
北京科技	北京科技	010-22222222	北京市	100001	删除
北京制药	北京制药	010-2233299	北京市	100001	删除
科技公司	科技公司				删除
制造厂	制造厂	010-22222233	北京市	100001	删除

图 12-17　用户管理页面

1. delusers.aspx 的 HTML 主要代码

在页面中使用了 GridView 控件，该控件增加了【删除】列，用于删除不需要的用户。代码如下：

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
…
<Columns>
  <asp:BoundField DataField="用户名" HeaderText="用户名" ReadOnly="True" />
  <asp:BoundField DataField="真实姓名" HeaderText="真实姓名" ReadOnly="True" />
  <asp:BoundField DataField="电话" HeaderText="电话" ReadOnly="True" />
  <asp:BoundField DataField="地址" HeaderText="地址" ReadOnly="True" />
  <asp:BoundField DataField="邮编" HeaderText="邮编" />
  <asp:CommandField ShowCancelButton="False" ShowDeleteButton="True" />
</Columns>
 …
</asp:GridView>
```

2. delusers.aspx.cs 的主要代码及其解释

(1) 每次加载时判断管理员是否已经登录。

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Session["admin"] == null)
```

```
    {
        Response.Redirect("login.aspx");
    }
    bindgrig();
}
```

(2) bindgrig()自定义函数，负责显示用户信息。

```
void bindgrig()
{
    string strsql = "select 用户名,真实姓名,电话,地址,邮编 from 用户 where 管理员标志=0";
    DataTable dt = BaseClass1.ReadTable(strsql);
    GridView1.DataSource = dt;
    GridView1.DataBind();
}
```

(3) 单击【删除】链接时触发 GridView1_RowDeleting 事件。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    //删除行处理
    String strsql = "delete from 用户 where 用户名='" +
                    GridView1.DataKeys[e.RowIndex].Value.ToString() + "'";
    BaseClass1.execsql(strsql);
    bindgrig();
}
```

12.3　编译与发布

网站或 Web 应用程序设计开发完成后，需要发布才能让用户访问。VWD 2008 提供了发布网站的功能，该功能将网站编译为一组可以通过 IIS 直接执行的文件，然后将这些文件复制到目标 Web 服务器上。

需要注意的是，VWD 2008 Express 速成版不具有发布的功能。虽然 VWD 2008 Express 速成版具有安装简便、运行速度快及适合初学者学习等特点，但在实际开发过程中，建议安装其他版本的 VWD 2008，比如，VS 2008 标准版、VS 2008 专业版以及 VS 2008 团队版等。

发布网站步骤如下：

(1) 在【解决方案资源管理器】中，用鼠标右键单击网站名，选择【发布网站】命令，在弹出的对话框中设置目标位置为 C:\xksoft，如图 12-18 所示。

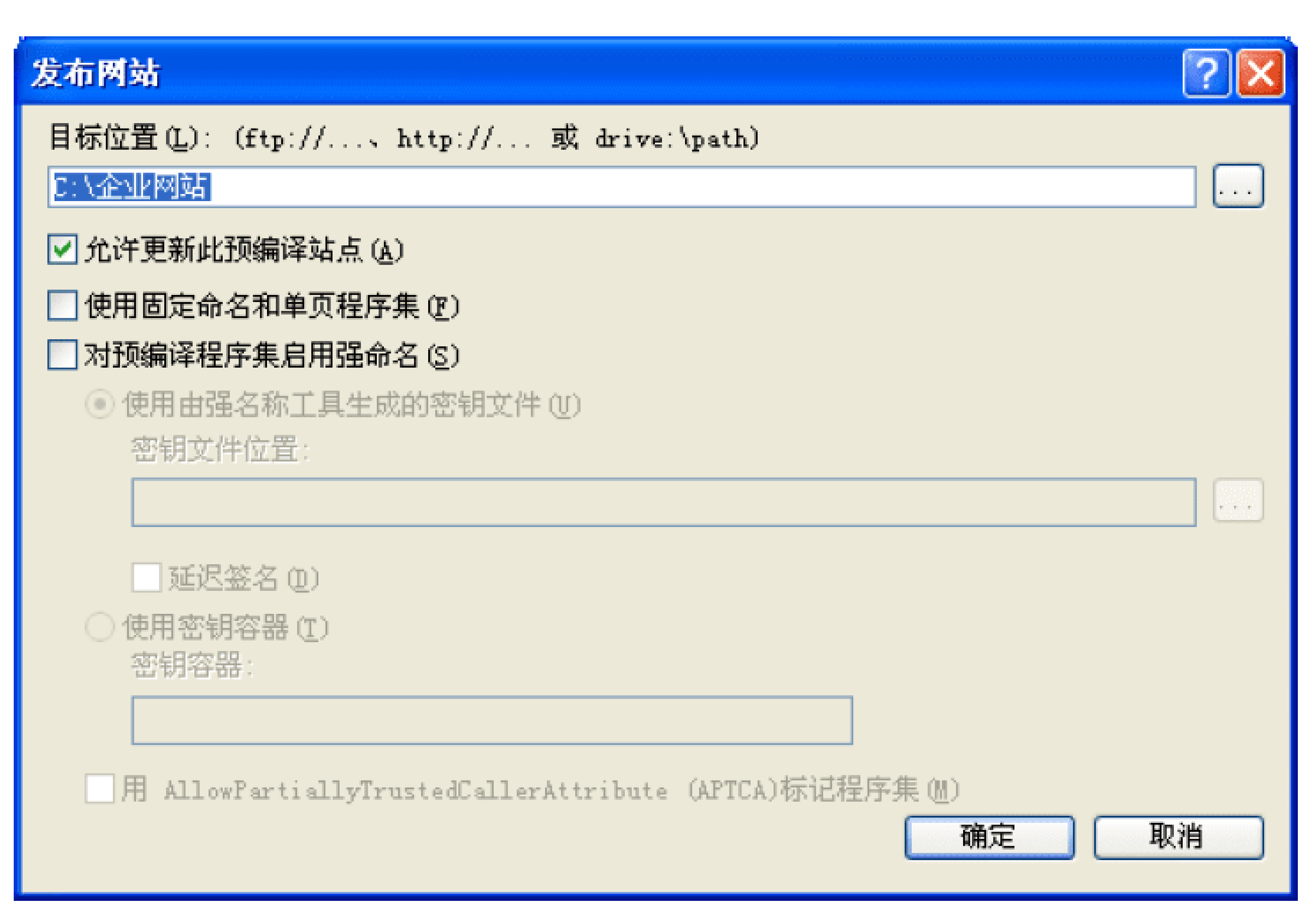


图 12-18　【发布网站】对话框

(2) 单击【确定】按钮，VWD 2008 编译网站，并将编译结果文件写入到指定的位置中。

(3) 通过【控制面板】|【管理工具】|【Internet 信息服务管理器】，打开 IIS 的设置窗口。

(4) 用鼠标右键单击【默认网站】，选择【新建】|【虚拟目录】命令，系统弹出“虚拟目录创建向导”对话框，输入虚拟目录别名为 xksoft，选择刚才设置的发布目标位置，配置完成后如图 12-19 所示。

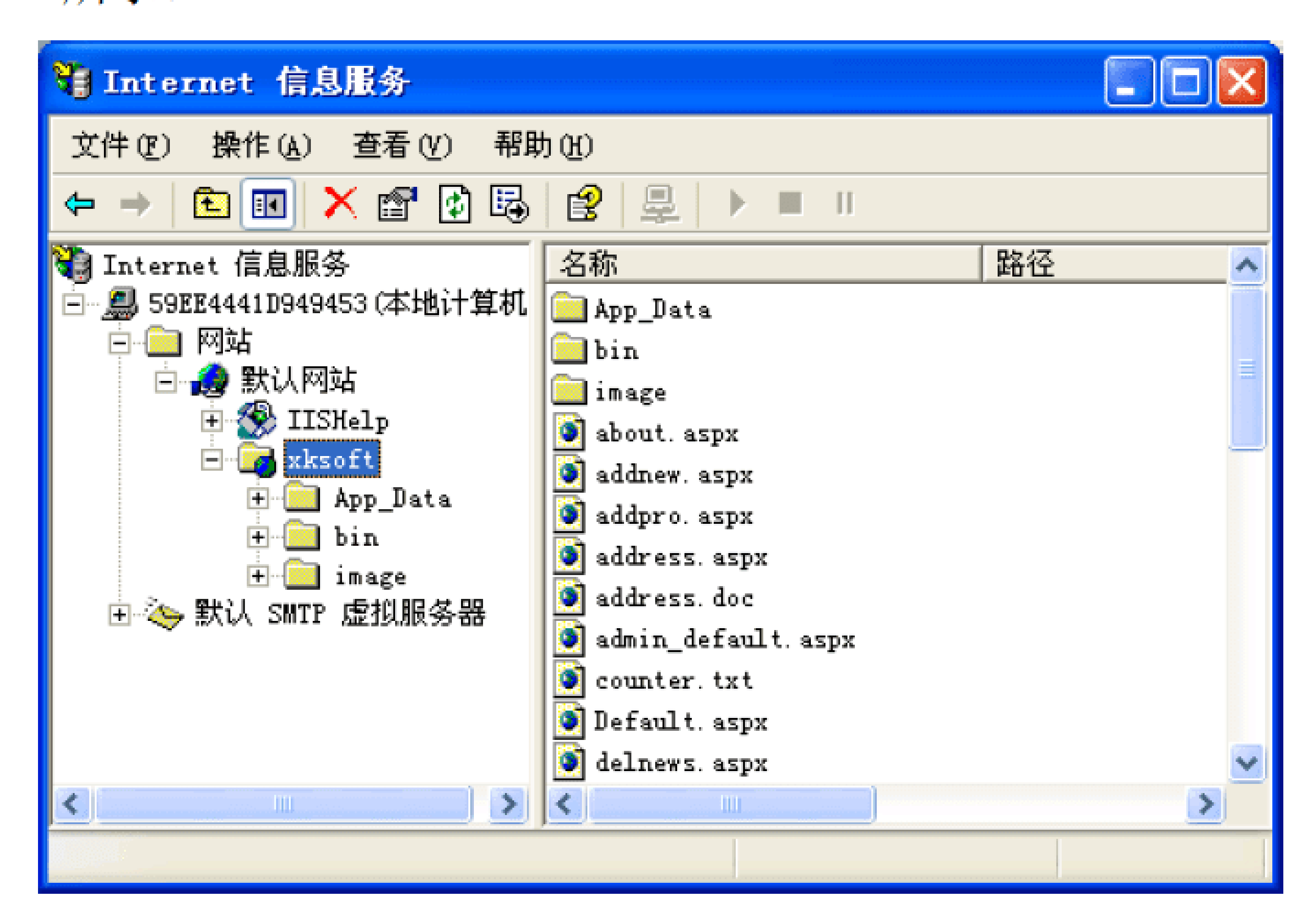


图 12-19　IIS 设置

(5) 打开浏览器，用 IE 访问 Http://localhost/xksoft/default.aspx，就可以看到如图 12-5 所示的效果。

如果需要将网站发布到互联网上，完成上面步骤(1)、(2)后，将编译后的文件上传到互联网空间就可以了。用户可以通过该空间绑定的域名访问网站。

实验 12　实例开发

1. 实验目的

通过实例练习，系统复习本书各章节的内容，掌握网站或 Web 应用程序的设计开发方法，提高开发水平。

2. 实验内容和要求

根据自己的兴趣设计开发一个网站，网站内容不限，可以是中小企业网站，班级网站，网上商店，网上书店，网上花店，也可以是展示自己的个人网站。无论选择什么样的内容，要求做到以下几点：

(1) 必须使用母版页。

(2) 应用 ASP.NET AJAX 无页面刷新技术。

(3) 使用数据库。

(4) 利用 GridView、DataList 控件，并有分页功能。

(5) 具有上传、下载文件的功能。

(6) 具有用户注册、登录的功能。

(7) 网页布局美观、色彩协调。

参 考 文 献

[1] 马骏，党兰学，杜莹. ASP.NET 网页设计与网站开发. 北京：人民邮电出版社，2007

[2] 金雪云. ASP.NET 简明教程(C#篇). 北京：清华大学出版社，2006

[3] 江红，余青松. 基于.NET 的 Web 数据库开发技术实践教程. 北京：清华大学出版社，2007

[4] 蒋培，王笑梅. ASP.NET Web 程序设计. 北京：清华大学出版社，2007

[5] 王成良. Web 开发技术及其应用. 北京：清华大学出版社，2007

[6] 朱晔. ASP.NET 第一步. 北京：清华大学出版社，2007

[7] 杨云，王毅. ASP.NET 2.0 程序开发详解. 北京：人民邮电出版社，2007

[8] [美]Scott Mitchell. ASP.NET 2.0 入门经典. 北京：人民邮电出版社，2007

[9] 程不功，龙跃进，卓琳. ASP.NET 2.0 动态网站开发教程. 北京：清华大学出版社，2007

[10] 王昊亮，李刚. Visual C# 程序设计教程. 北京：清华大学出版社，2003

[11] 张克凡，陈立伟，刘伟光. ASP.NET 网络程序设计及应用. 北京：航空航天大学出版社，2007

[12] 肖建. ASP.NET 编程基础. 北京：清华大学出版社，2004

[13] 杜亮. 亲密接触 ASP.NET. 北京：清华大学出版社，2002

[14] 冯方等. ASP.NET 上机练习与提高. 北京：清华大学出版社，2007

[15] 王军，郭卫勇. ASP.NET2.0 大揭秘. 北京：清华大学出版社，2004

[16] 张英男，高喆，林丁报，靳海江. ASP.NET 网络编程学习笔记. 北京：电子工业出版社，2008

[17] 王洪超. 新概念 ASP.NET 教程. 北京：科海出版社，2008

[18] 雷之宇，朱冰，周烨等. C#语言实用教程. 北京：铁道出版社，2004

[19] 王愉. DHTML 动态网页设计 北京：人民邮电出版社，2007

[20] www.msdn.com

ASP.NET 3.5网站开发实例教程

主要内容

本书全面讲述了ASP.NET 3.5网站开发方法和Web应用程序的技术。主要内容包括：ASP.NET与VWD 2008开发平台，XHTML基础知识，ASP.NET基础知识，ASP.NET服务器控件，页面切换与导航，页面布局，ADO.NET数据访问，ASP.NET中的数据绑定，ASP.NET AJAX，Web服务，在ASP.NET中使用XML等，并且运用大量实例对各种关键技术进行了深入浅出的分析。

读者对象

本书可作为高等院校ASP.NET开发课程的教材，还可作为有一定的面向对象编程基础和数据库基础，想利用VS 2008开发Web应用程序的开发人员的参考资料。

本书特色

本书内容丰富，结构合理，实例翔实，浅显易懂。在每一章的正文中，结合所讲述的关键技术，穿插了大量极富实用价值的实例。末尾安排了有针对性的实验，增强对基本概念的理解和实际应用能力。

本书配有电子教案和所有实例与实验的源代码，既便于教师教学，也便于学生学习，具有很强的实用性。

978-7-302-19617-4

定价：30.00元

本书电子教案、实例源文件和实验答案可通过
http://www.tupwk.com.cn/downpage网站免费下载。